
NSD Smart Display

User's Manual and Software Developer's Guide



RELEASE NOTES

Version	Date	NOTES
1.00	April, 2012	Initial release

Disclaimer

This documentation is provided for use with IC Nexus products. No license to IC Nexus property right is granted. IC Nexus assumes no liability, provides no warranty either expressed or implied relating to the usage, or intellectual property right infringement that may result from its use.

IC Nexus provides this document “as is,” without warranty of any kind, either expressed or implied, including, but not limited to, its particular purpose. IC Nexus may make changes to this document without notice.

Table of Contents

Chapter 1: Introduction	4
Chapter 2: Development Environment Setup.....	5
Eclipse Download and Installation Instructions	5
Android SDK Starter Package	12
Android Debug Bridge.....	24
Android Development Toolkit.....	26
AVD configuration	30
Chapter 3: Guide to Build a Sample Application on NSD	33
Sample Digital Signage.....	33
SampleChart	40
SampleUserDragAndDrop	46
Chapter 4: Example Applications on NSD	52
Chapter 5: Remote Control.....	66
Appendix A: NSD Software SDK/API	73
Elevator	73
Graphical Drawing UI	78
Application with Drag and Drop UI.....	82
Digital Signage	88
Chart.....	98
MediaPlayer API.....	104
ResourceUsage API.....	109
Appendix B: NSD API Categories	117
Android apk and jar files.....	117
Events.....	117
Services.....	118
Sending and Receiving Commands via the Network	119
NSD Command Definition	120

Chapter 1: Introduction

The NSD smart display is a combination of three components: a high resolution TFT LCD module with touch panel, a high-performance ARM1176 processor and an Android operating system.

NSD can work as a display device with various terminal ports, such as low speed RS232, RS485 and high speed USB, to process commands from an 8-bit MCU or 32-bit CPU. The commands include launching applications in NSD to run texts, images, videos and 3D animation display tasks. This feature helps product developers to rapidly integrate this TFT LCD module to their equipment without modifying or upgrading hardware and processor. Typical applications include an elevator display, home electronics touch control panel, industrial equipment display, rack-mount server display, and exercise equipment display.

NSD is also capable of running as a standalone Android device with a touch screen LCD and high-performance ARM processor. Typical application usages include digital signage devices, multimedia players, POS system, vending machines, home automation console and vehicle infotainment system.

Since NSD is an Android-based open system, many development tools and open source resources are available to use for product development. Developers can obtain a huge amount of free resources to enable them quickly complete the product design. This makes NSD superior to other competitors, who use proprietary systems.

The NSD series is available with display sizes ranging from 3.5" to 10.1" LCD. Other LCD sizes can also be made available with prior customer requirements.

This manual is intended to be used as a practical and informative guide only for the NSD smart display, from the installation to the operation of the product. If you have questions, comments, or concerns about your NSP smart display, please let us know at sales@icnexus.com.tw

Thank you for choosing the NSD Smart Display.

Chapter 2: Development Environment Setup

Eclipse Download and Installation Instructions

If you would like to download the latest Eclipse version, please visit the Eclipse official website <http://www.eclipse.org/> and click the “Downloads” link on the main navigation bar. The downloaded file comes in a zip file package.

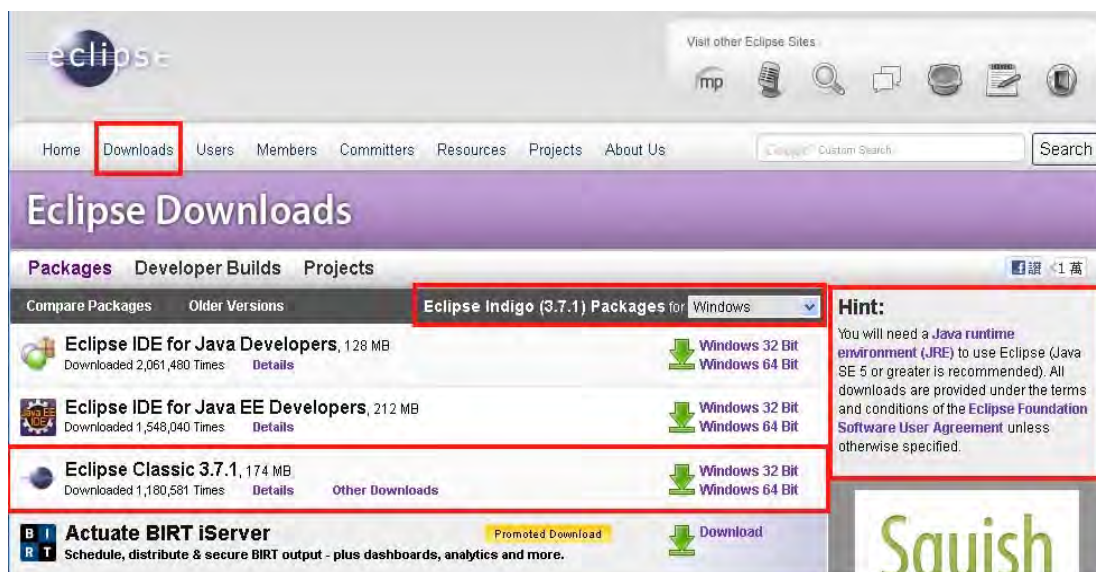


Figure 2-1: The download page

To save your download time, a version of Eclipse is included in the NSD software DVD. In the DVD, locate the folder with “Eclipse” name and change to the folder that matches operating system (Windows or Linux) and CPU (x86 for 32-bit and x64 for 64-bit CPU) of your development PC.

After unpacking the contents and entering the Eclipse folder, you will see an executable file called Eclipse.exe in Windows (Note: There are no filename extensions in Linux). You should be able to start the program by double-clicking the Eclipse.exe file.

If an error message “Java not installed” appears while you attempt to start the program, please perform a self-installation on the latest JRE (Java Runtime Environment) and JDK (Java Development Kit). You can find a direct link to download Java at the Hints section on the right hand side of the Downloads Page (Figure 2-1).

Linux users can follow the steps below to install SUN JRE:

```
$ sudo add-apt-repository ppa:sun-java-community-team/sun-java6
$ sudo apt-get update
$ sudo apt-get install sun-java6-jre sun-java6-bin sun-java6-jdk
```

Seeing that this program only supports 32-bit operating system or a single core processor, for all 64-bit system users (most dual core processors work on 64-bit operating system), please download the following package to ensure successful execution:

```
$ sudo apt-get install ia32-libs
```

Running Eclipse

When you run Eclipse the first time you will get a dialog box like the one shown in the figure below, prompting you to choose a workspace location for storing your project files.

Click “OK” after you have selected your preferred workspace location to store your projects. Now sit back and wait for the loading process to complete.

If you do not wish to be asked each time the program starts, you can check the box next to “Use This As The Default And Do Not Ask Again” and click “OK”. (Figure 2-2)

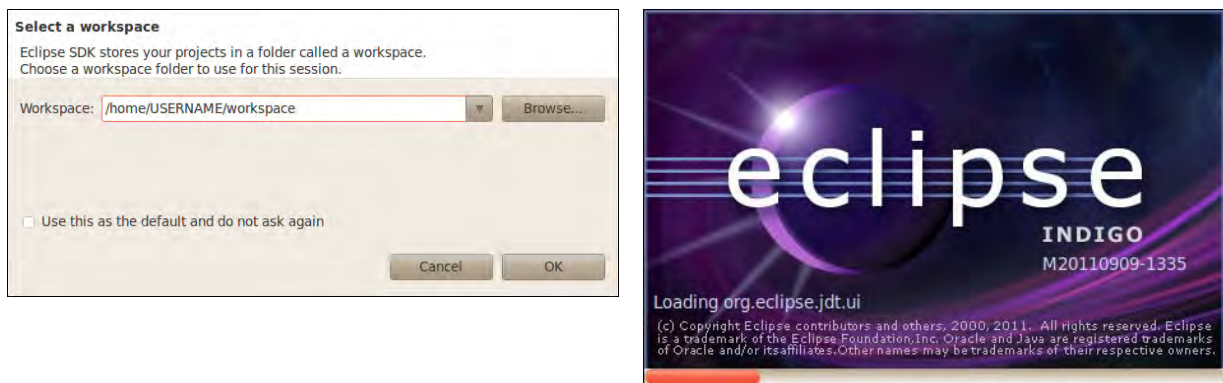


Figure 2-2: The workspace launcher dialog box and Eclipse loading box

Eclipse Setup for Android Development

(Please refer to the tips at the end of this section for common installation issues).

Open Eclipse, then go to [Help] Menu → Select [Install New Software]. This will open the Eclipse installer dialog box shown in the figure 2-3.

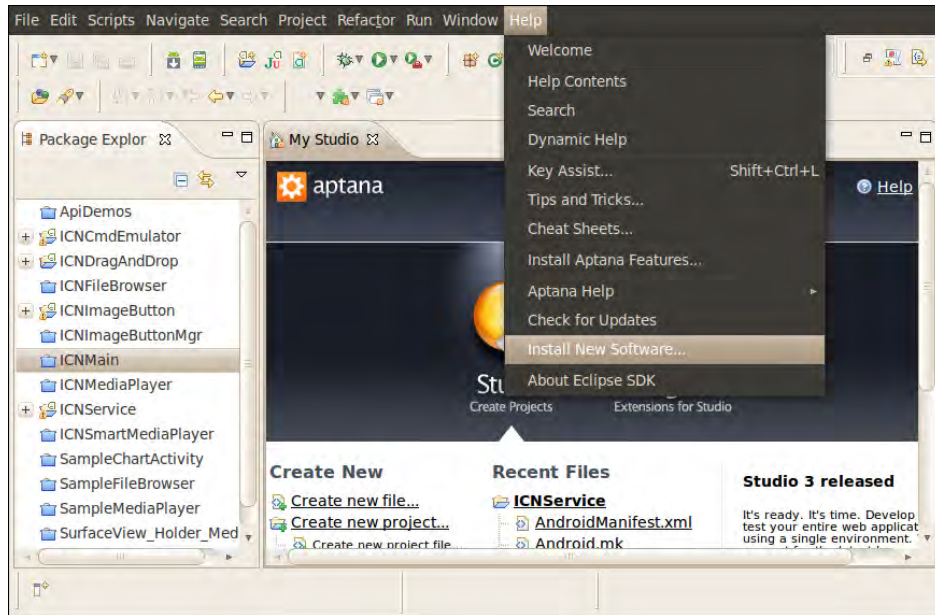


Figure 2-3: Select “install new software...”

Step 1: Install Android Development Tools

To add a new repository, you can just type the name and the URL for the location of Android Development Tools (ADT) in the text box labeled Work With and the Eclipse installer will go out to that address to look for add-on packages. The name and the URL for the ADT (reference only) to be entered are:

Android ADT - <https://dl-ssl.google.com/android/eclipse>

(A custom repository name to be used on your computing machine) – (the location of the Internet resource)

In just a few seconds the Eclipse installer will determine if there is available software of interest to you. Once found, unfold the Developer Tools update entry. You will be presented with all the package names and their version numbers that are available for download. Select all the items underneath the Developer Tools line and then click [Next]. You may check the [Hide items that are already installed] option to remove those items from the list that are already installed. The remaining items will be new ones or that those have not yet been installed (Figure 2-4).

The Eclipse installation system will now attempt to retrieve information and check for updates of all of those Android plug-in dependencies. This may take a while depending on your Eclipse environment so please be patient.

Go to “Next” → Read the license agreement → Select “I accept the terms of the license agreements” → Click “Finish” to begin the installation process.

If a security warning message appears, click “Yes” to continue with the installation.

Once the download has finished, you will be instructed to restart Eclipse for the changes to take effect. Click “Yes”.

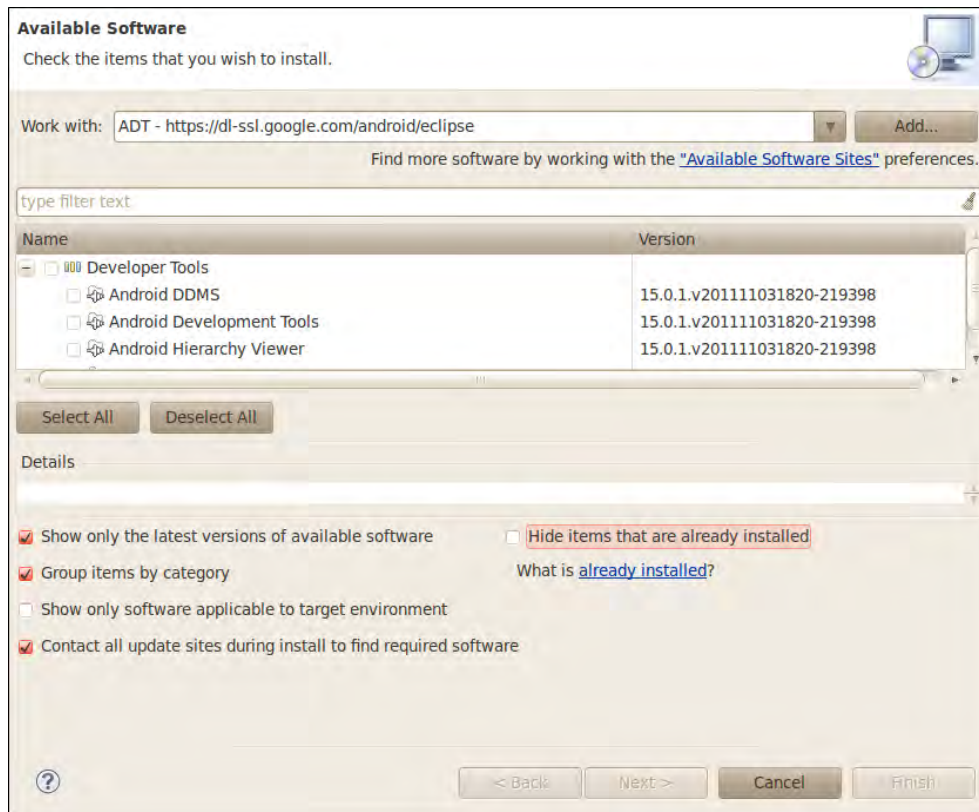


Figure 2-4: ADT

After you restart Eclipse, a pop-up window will appear asking you to install the SDK to develop for Android.

Select 'Install new SDK' → click "Install the latest available version of Android APIs (support all the latest features)" → 'Next' → 'Finish'

Note:

If you receive a message like "Do you want to download/install Android SDK?" please choose "No". You should follow the steps below to install Android SDK by manually way (chapter: Android SDK Starter Package).

Click the "Window" option on the top-menu bar → Click "Preferences" → Choose the "Android" from the left table → Edit the text of "SDK Location:" or click the "Browse..." to find the Android SDK folder by file-browser.

If the setting path is correct, you should see information something like figure 2-5.

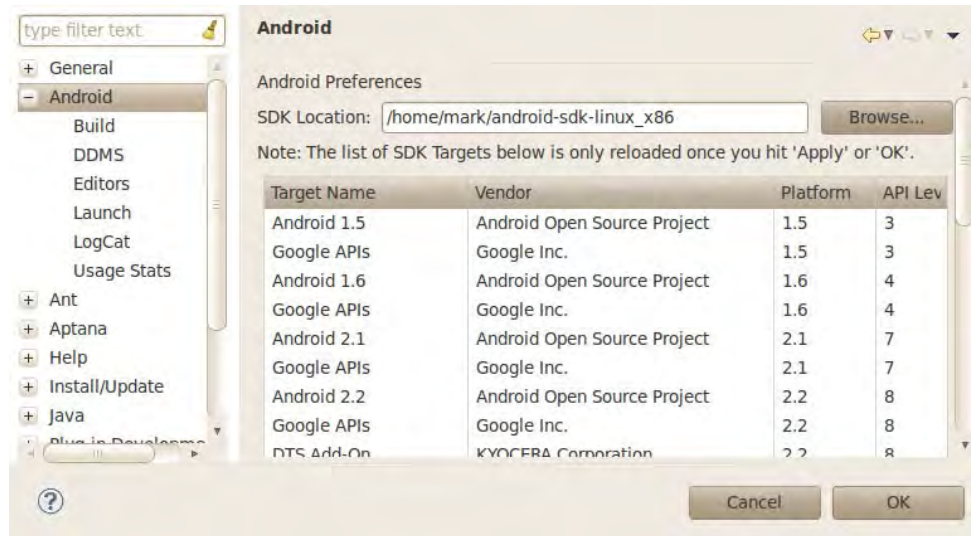


Figure 2-5: ADT

Step 2: Install Aptana Studio

In order to edit CSS, JavaScript and HTML, you will need Aptana Studio. Applying the same procedure as how ADT was installed, the name and the URL for the Aptana Studio (reference only) to be entered are:

Aptana Studio - <http://download.aptana.com/tools/studio/plugin/install/studio>

After this download, you can check for other updates and the add-ons to Eclipse from the Help menu at regular intervals.

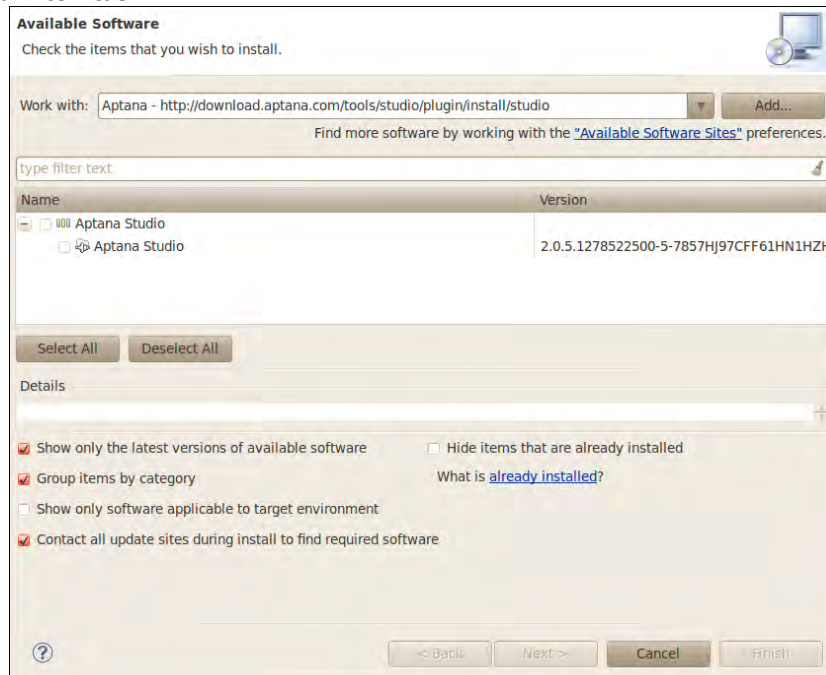


Figure 2-6: Aptana

Step 3:

The steps to update Eclipse are very similar to those described in the previous paragraphs. The only difference is the name and location where you obtain the update:

The Eclipse Project Updates - <http://download.eclipse.org/eclipse/updates/3.7>

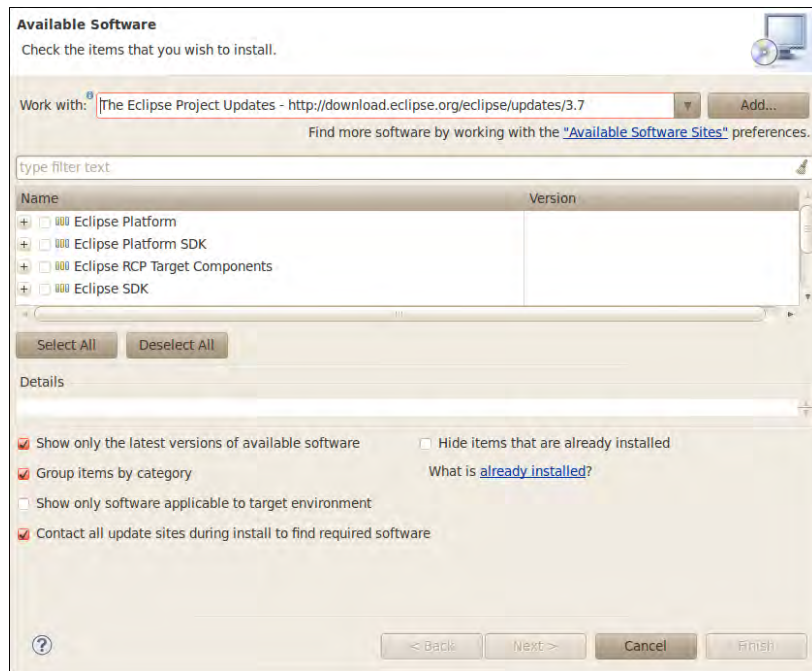


Figure 2-7: Eclipse update

Tips:

What is already installed on Eclipse or has been consulted for updating the Eclipse will all be recorded in the [Available Software Sites] page.

Go to [Window] menu → select [Preferences] (Figure 2-8) → click [Install/Update] → [Available Software Sites] (Figure 2-9).

To add/remove sites from the system, users can click [Add] and [Remove]. To edit the sites, apply the [Edit] button. For other usage, please refer to the documents on the Eclipse official website.

Press the [Add] button to specify a custom name and enter a location of a new site.

If you are unable to add a new site successfully in the [Install New Software] wizard, you can update the location in the [Available Software Sites] page instead before installing the plug-in in the [Install New Software] wizard.

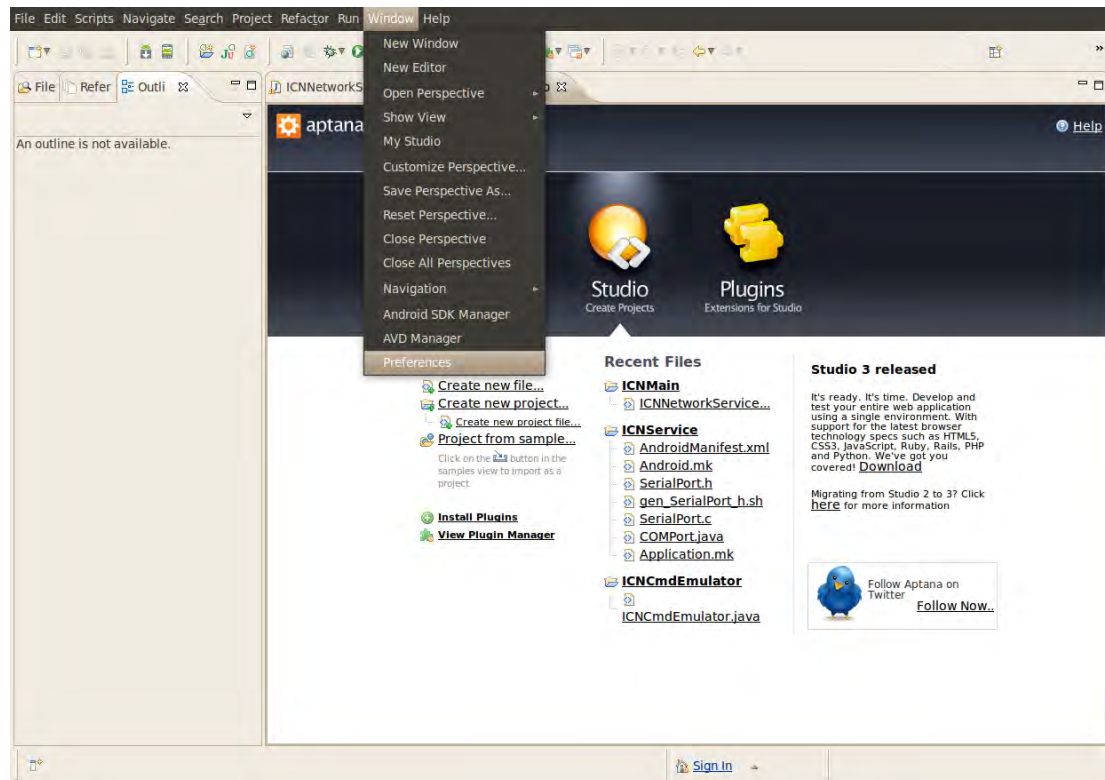


Figure 2-8: Open window preferences

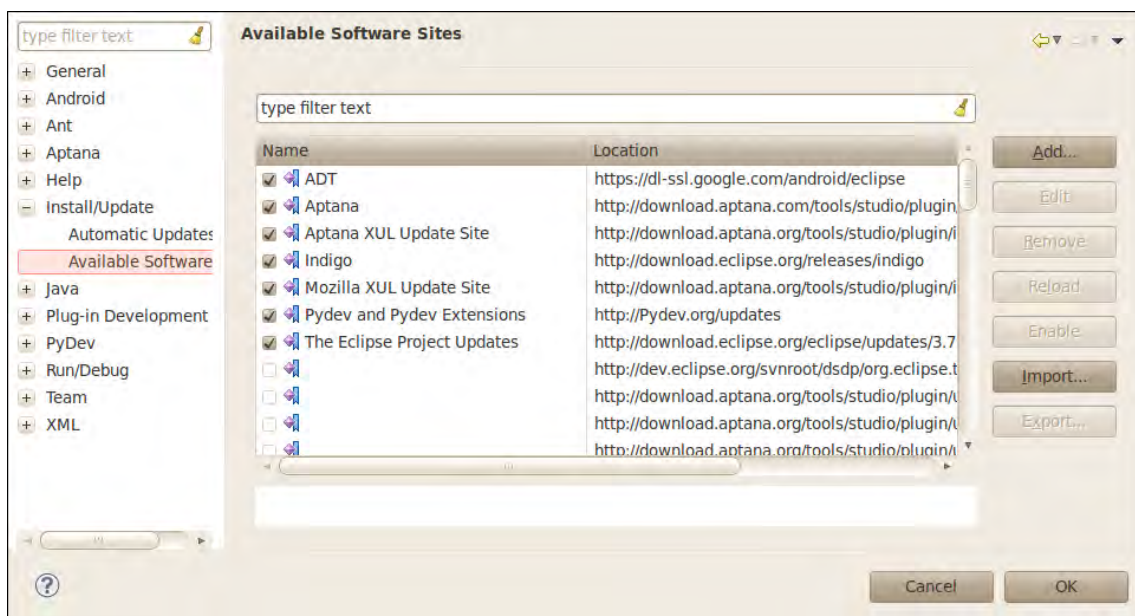


Figure 2-9: Available software sites

Android SDK Starter Package

Please visit the following site to download the latest tools to perform Android SDK.

<http://developer.android.com/sdk/index.html>

To save your download time, a version of Android SDK is included in the NSD software DVD. In the DVD, locate the folder with “Android SDK” name and change to the folder that matches operating system (Windows or Linux) of your development PC.

Uncompress the zip files after the download completes (See pictures below, Figure 2-10).

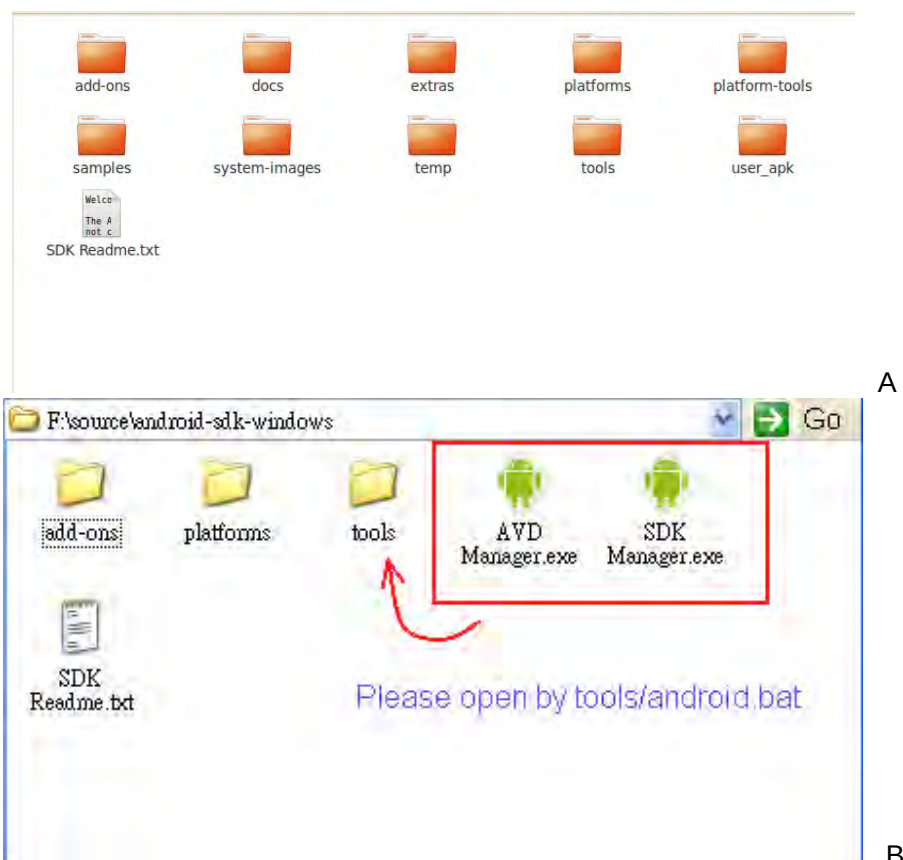


Figure 2-10: Linux (A) and Window (B) source folder snap-shoots

Step 1:

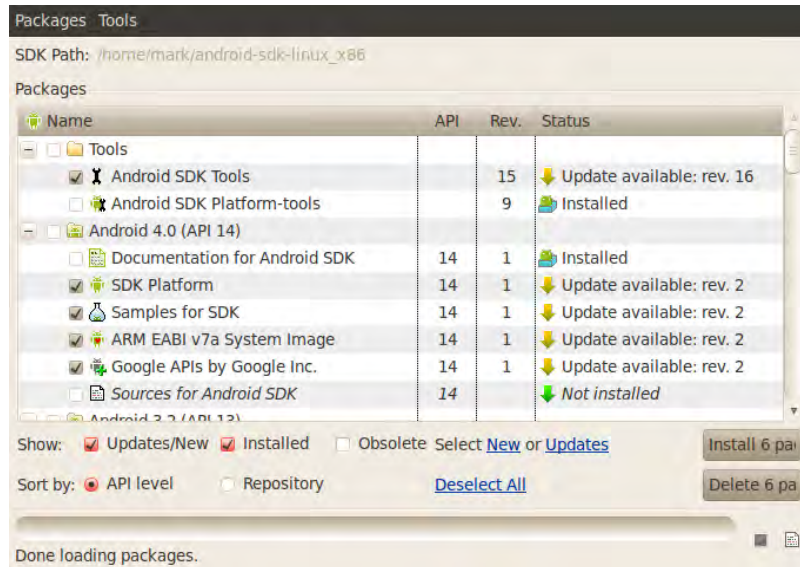
Open up the Android SDK directory → The [Tools] folder → Click the android.bat file to run the Android SDK Manager (For **Linux users**, please use Script file name android).

You may use the Android SDK Manager to get the updates of the existing installed SDK packages. (Figure 2-11)

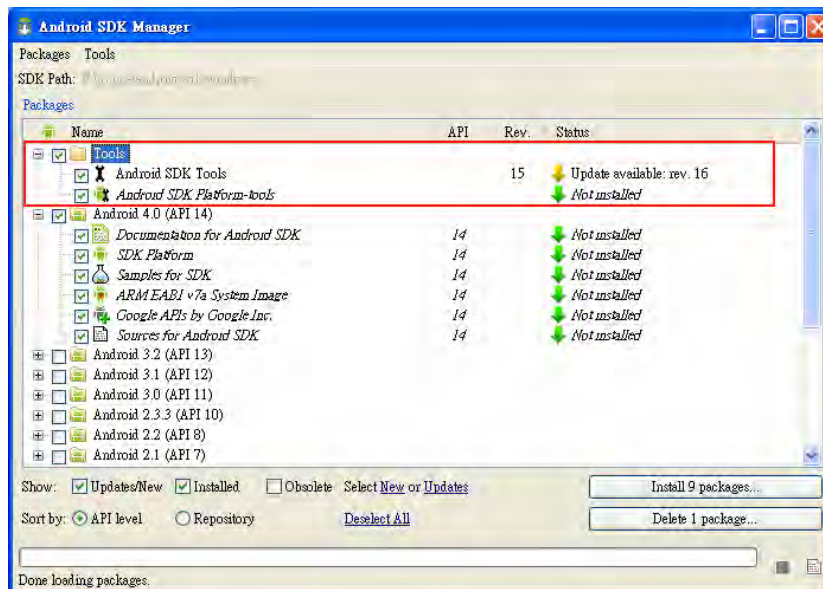
Step 2:

The AVD manager is an easy way to create an Android Virtual Device (AVD). No matter if it is a Linux or Windows OS, you will need to download the Android API and update the latest SDK tools for the first time. It is advised to install all the updates for the benefit to the development of Android as whole.

Please be patient to wait for all the download tasks to finish. This process will depend on your internet speed.



Android SDK (Linux)



Android SDK (Windows)

Figure 2-11: different Android SDK Manager in different operating system

Step 3:

You could update the environment variables to make tool access easier by adding the location of the SDK's tools and platform-tools to your path.

Linux :

After the Android SDK has installed, users can create a variable in the `.bashrc` file in their home directory. For example:

```
export PATH=$PATH:~/android_sdk/platform-tools:~/android_sdk/tools
```

Windows:

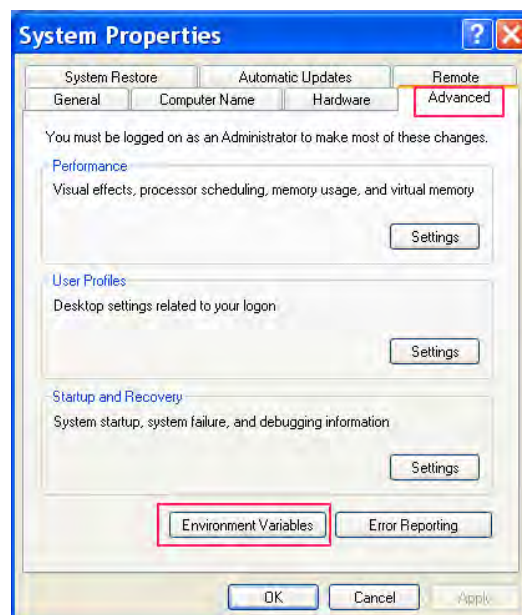
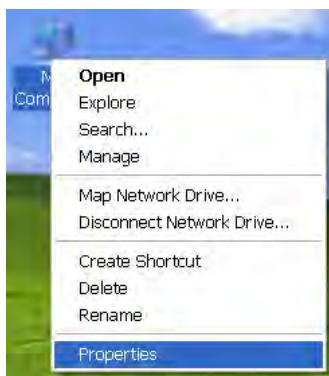
Right-click [My Computer] → click Properties → Click on the Advanced tab → Click [Environment variables].

The Environment variables dialog box lists two kinds of variables – the upper one is for the current user setting and the lower one applies to the whole system. Add the system variables only if you do not know what you are up against. (Figure 2-13)

Locate the Path variable → Double-click "Path" or click "Edit" to change its value to:

```
;D:\android-sdk-windows\tools;D:\android-sdk-windows\platform-tools
```

After you have done that, start Eclipse SDK.



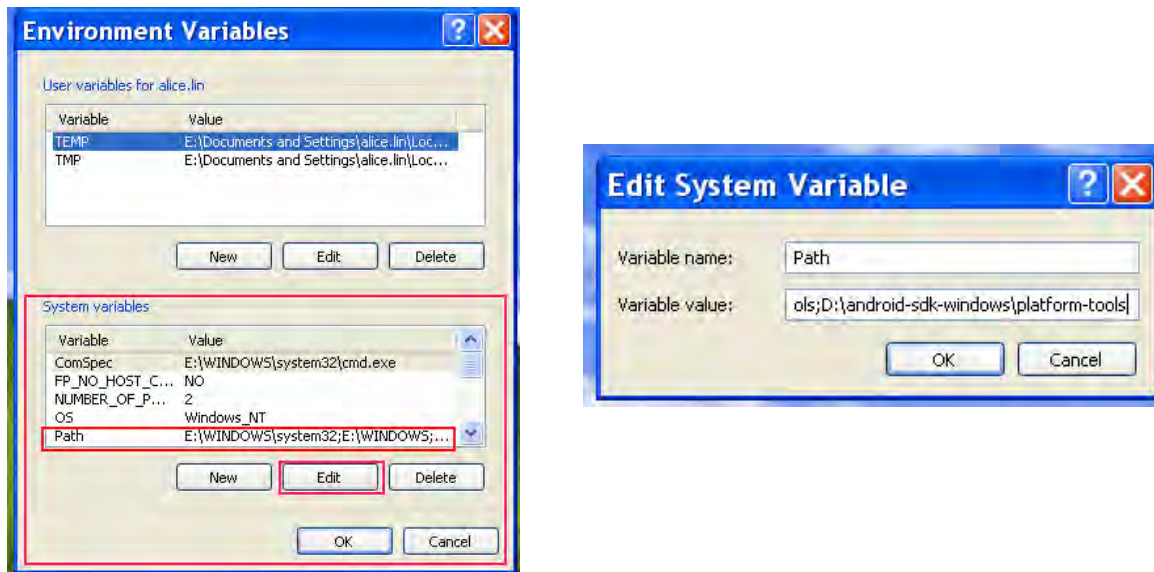


Figure 2-13 : Add Android SDK path into a Windows XP

Installing ADB and a USB Driver for an Android device

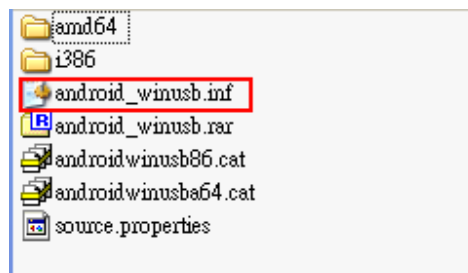
NSD has specific set of architecture views, so please follow the steps below to guide you through the installation process.

For Windows operating environment: (in Windows XP SP3)

1. Make sure your PC already downloaded the Google USB Driver package using the Android SDK Manager.



2. In the `...\android-sdk-windows\extras\google\usb_driver` folder, you can find the content like the one below.



Note: You must replace the `android_winusb.inf` with the NSD `android_winusb.inf`.

3. Connecting the mini-usb cable to your PC. Now you can turn on the NSD power.

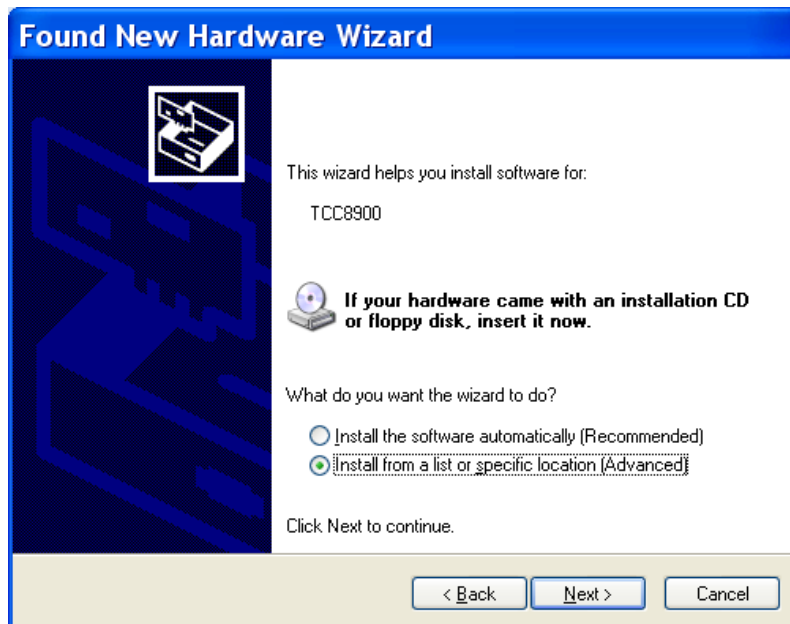


4. Please follow the steps below to download the Android ADB Interface.

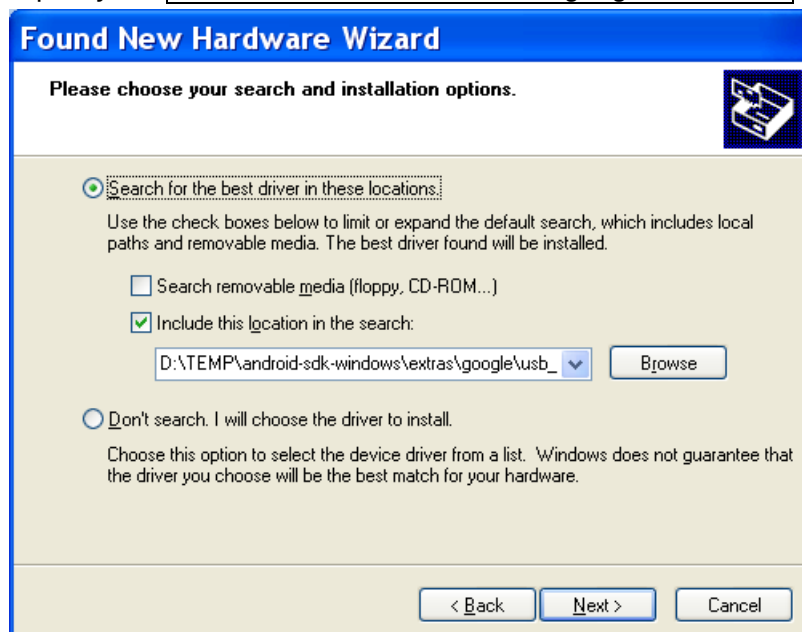
The “Found New Hardware Wizard” will be displayed and will prompt you to install software for the Android ADB Interface. Choose ‘No, not this time’ → Click ‘Next’ to continue.



Select “Install from a list or specific location (Advanced)” → Click ‘Next’ to continue.



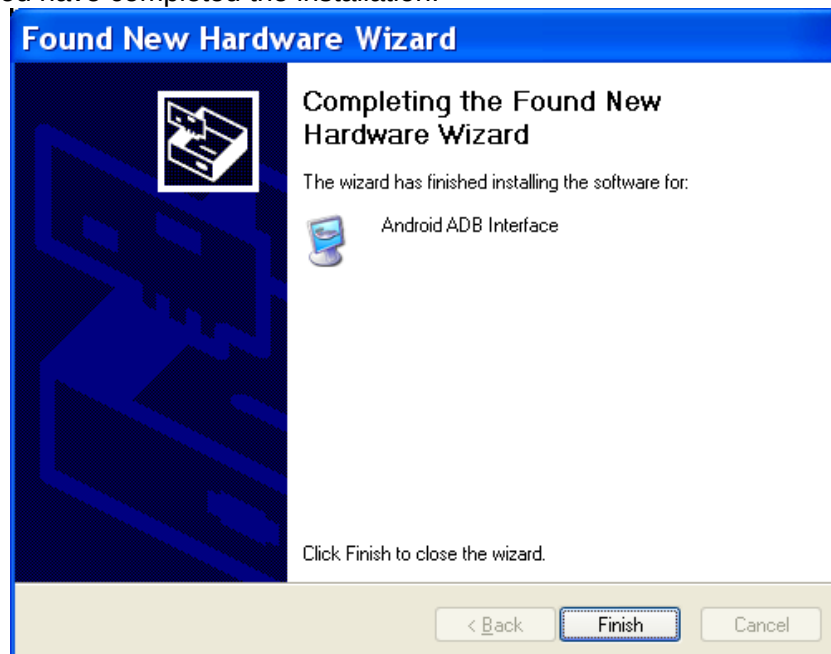
Select “Search for the best driver in these locations” → Check “Include this in the search” → Click ‘Browse’ to specify the `...\android-sdk-windows\extras\google\usb_driver` folder.



Please wait for the software to be installed.



At this point, you have completed the installation.

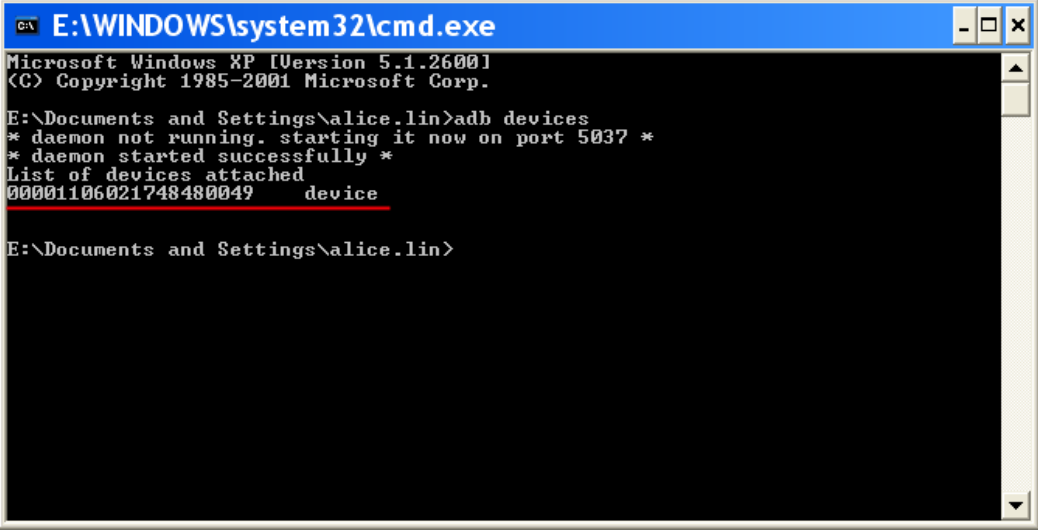


Right-click my Computer icon → Properties → Hardware → Select 'Device Manager' → Open up 'Android Phone' to see if there is an 'Android ADB Interface'.



Then, you can test the Android *adb* tools in the command prompt window, using the command below to check if your daemon has been started successfully.

> adb devices



```
C:\ E:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

E:\Documents and Settings\alice.lin>adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
00001106021748480049    device

E:\Documents and Settings\alice.lin>
```

For Linux users: (in Ubuntu 10.04 Lucid Lynx)

1. Connect the mini-usb cable to your PC. Now turn on the NSD power.



2. Open the terminator and type the command "lsusb" to check if there is any new device.

```
/opt/armv6/codesourcery/bin/:/opt/toolchain/mipsel-linux/bin/:/opt/eclipse/:/usr
/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/home/mark/a
ndroid-sdk-linux_x86/platform-tools:/home/mark/android-sdk-linux_x86/tools
mark@mark-ubuntu10:~$ lsusb
Bus 005 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 003 Device 003: ID 046d:c016 Logitech, Inc. M-UV69a/HP M-UV96 Optical Wheel
Mouse
Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 002 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
Bus 001 Device 019: ID 18d1:deed
Bus 001 Device 003: ID 1307:0361 Transcend Information, Inc.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
mark@mark-ubuntu10:~$
```

3. You can see the new USB device with an ID: **18d1:deed**.

Open the file name 50-android.rules. (If the file does not exist, just follow the step below. Let's create a new one.)

```
cd /etc/udev/rules.d
sudo gedit 50-android.rules
```

Add the `SUBSYSTEM=="usb", SYSFS{idVendor}=="18d1", MODE="0666"` into the file.



Save this and then exit.

4. Restart the Service

Please Type the following commands:

```
sudo chmod a+rx /etc/udev/rules.d/50-android.rules
sudo /etc/init.d/udev restart
cd [Android SDK Install location]/android_linux_sdk/platform-
tools/
sudo ./adb kill-server
```

5. Check

```
sudo ./adb devices
```

At this point you can use `adb devices` to check if your advice is attached to your PC successfully.

A terminal window screenshot showing the command `adb devices` being executed. The output is: `List of devices attached` followed by a line `00001111081441220084 device` which is underlined. The prompt is `mark@mark-ubuntu10:~$`.

```
/opt/armv6/codesourcery/bin:/opt/toolchain/mipse
/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/s
ndroid-sdk-linux_x86/platform-tools:/home/mark/a
mark@mark-ubuntu10:~$ adb devices
List of devices attached
00001111081441220084 device
mark@mark-ubuntu10:~$
```

Android Debug Bridge

Found in the Android SDK, the Android Debug Bridge (ADB) is a powerful command-line utility intended for developers to manage either an emulator instance or connected Android-powered device. It is a client-server program, which consists of three components: a client, a daemon and a server. The ADB server takes care of all the communication needed between the client running on the development machine and a daemon running on either an emulator or a target device. This allows developers to move files from their local machines to the Android devices based on the commands sent. The table below lists the command-line options which can be used to instruct ADB how to execute a specific task.

List of ADB command-line options

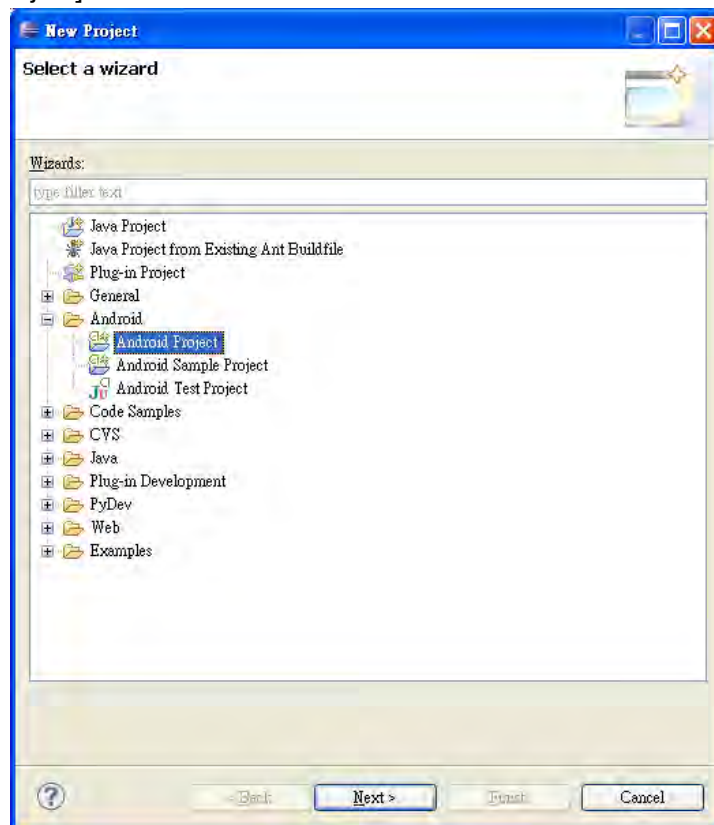
<code>adb push <local> <remote></code>	Copy file/dir to device
<code>adb pull <remote> [<local>]</code>	Copy file/dir from device
<code>adb sync [<directory>]</code>	Copy host->device only if changed (-l means list but don't copy)
<code>adb shell</code>	Run remote shell interactively
<code>adb push <local> <remote></code>	Copy file/dir to device
<code>adb shell <command></code>	Run remote shell command
<code>adb emu <command></code>	Run emulator console command
<code>adb logcat [<filter-spec>]</code>	View device log
<code>adb forward <local> <remote></code>	Forward socket connections Forward specs are one of the following: tcp:<port> localabstract:<unix domain socket name> localreserved:<unix domain socket name> localfilesystem:<unix domain socket name> dev:<character device name> jdwp:<process pid> (remote only)

<code>adb jdwp</code>	List PIDs of processes hosting a JDWP transport
<code>adb install [-l] [-r] [-s] <file></code>	Push this package file to the device and install it (-l means forward-lock the app) (-r means reinstall the app, keeping its data) (-s means install on SD card instead of internal storage)
<code>adb uninstall [-k] <package></code>	Remove this app package from the device (-k means keep the data and cache directories)
<code>adb bugreport</code>	Return all information from the device that should be included in a bug report
<code>adb help</code>	Show this help message
<code>adb version</code>	Show version number

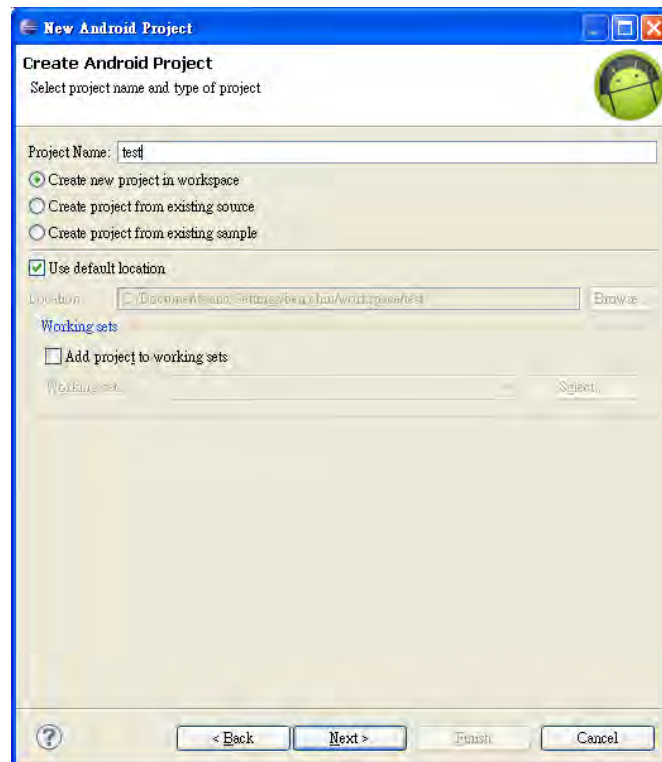
Android Development Toolkit

An Android project is designed as a library project, which acts a container for your application's source code, resource files and files. During the build process, your Android project will eventually be compiled and packaged into an .apk file so that you install onto a device.

To create a new Android project, go to [File] -> [New]->[Project...]-> Open the [Android] Folder -> Select [Android Project] -> Click the 'Next' button.



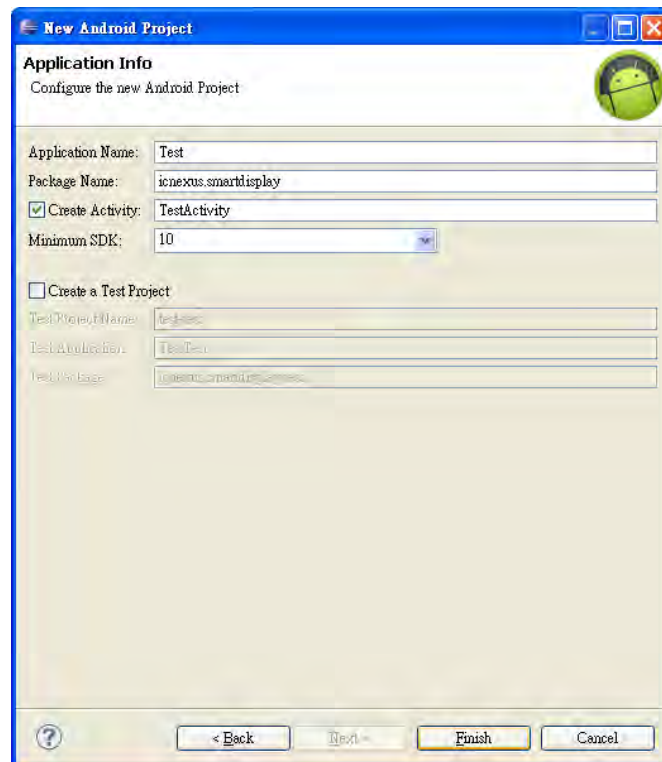
Case1: Create new project in workspace.



Select "Create new project in workspace" and input a project name.

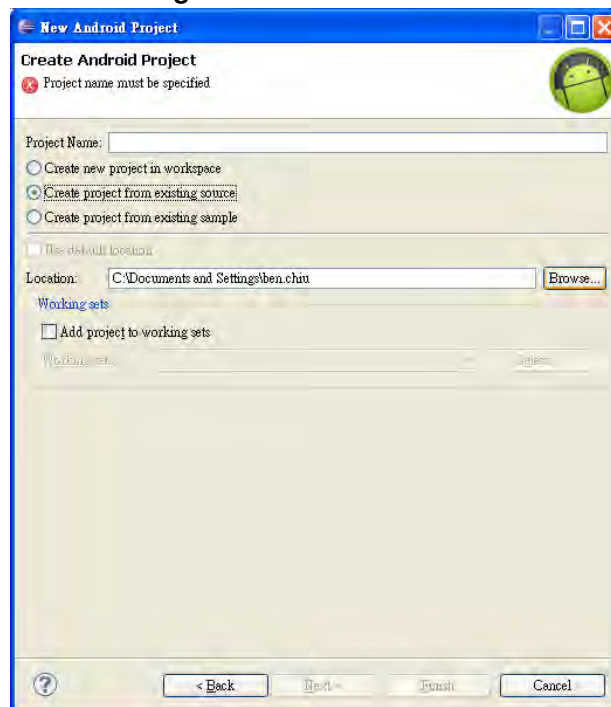


Select 'Android 2.3.3' and click 'Next'.

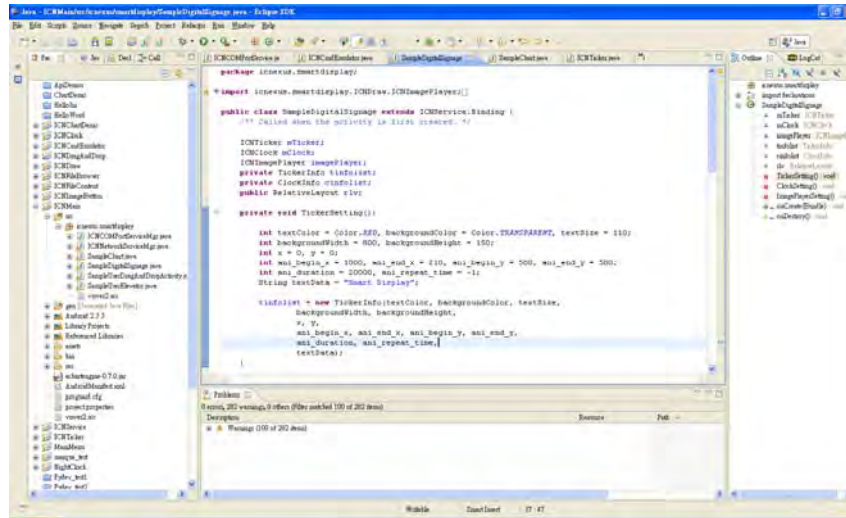


Input 'Package Name' and click 'Finish'.

Case2: Create project from existing source.



Select “Create project from existing source” and then click the “Browse” button to select an existing project file.

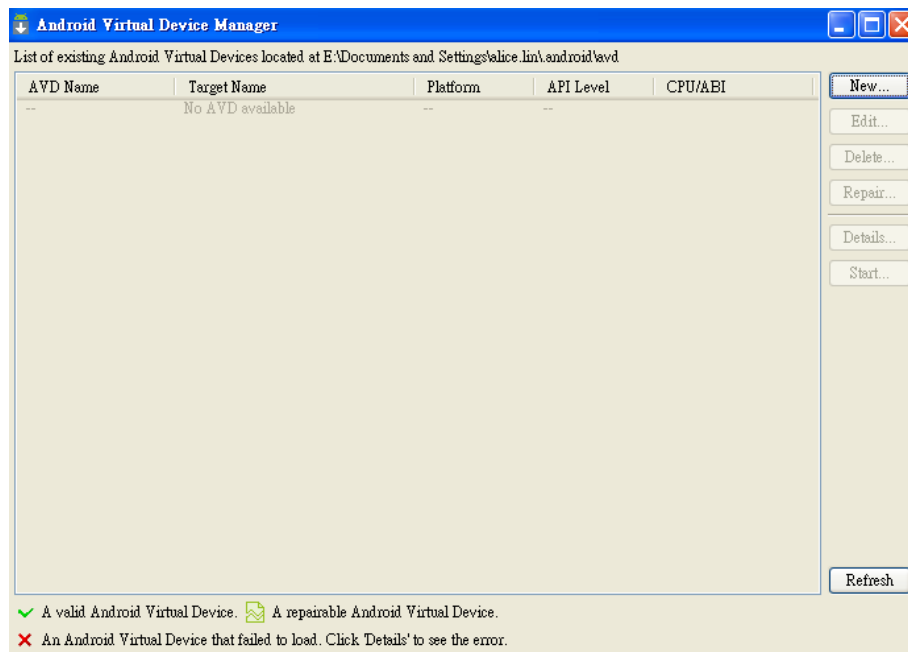


Once the ‘Finish’ button is clicked, the New Android Project wizard will create a new project for you. Now you can click the Run icon for the Eclipse to go through the compile-and deploy process. The Eclipse will then launch an emulator, opening up a window showing a mock-up of an Android mobile device if the emulator has already been configured on your development machine. If it has not, please refer to the following step to install your first Android Virtual Device (AVD).

AVD configuration

An Android Virtual Device (AVD) is an emulator that lets you create a virtual output of the Android mobile interface by defining its hardware and software options. The easiest way to configure your AVD is to use the AVD manager.

To start the AVD manager, you can either go to Window → AVD Manager in Eclipse or run the *android* tool on the command line.

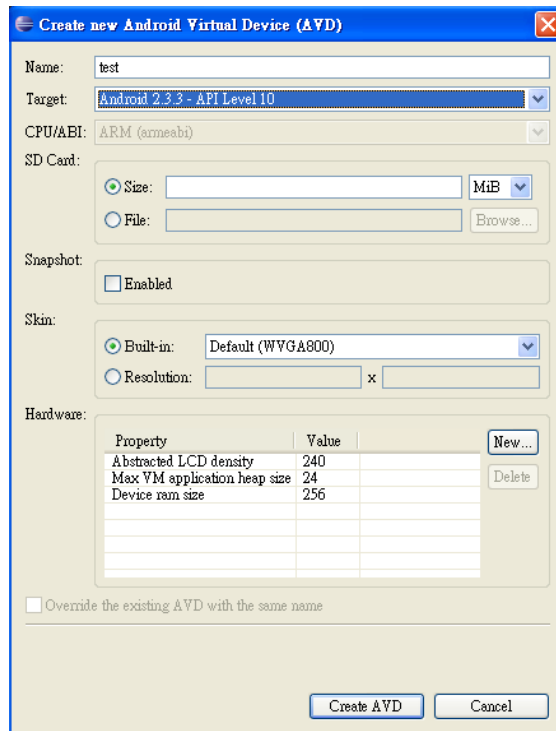


The AVD Manager

You are free to create as many AVDs as you would like to test on, but it is recommended that you should choose a higher API level to test on than the target API level for your application. Please note that it is impossible to run the API level that is lower than that required by your target application on a system image.

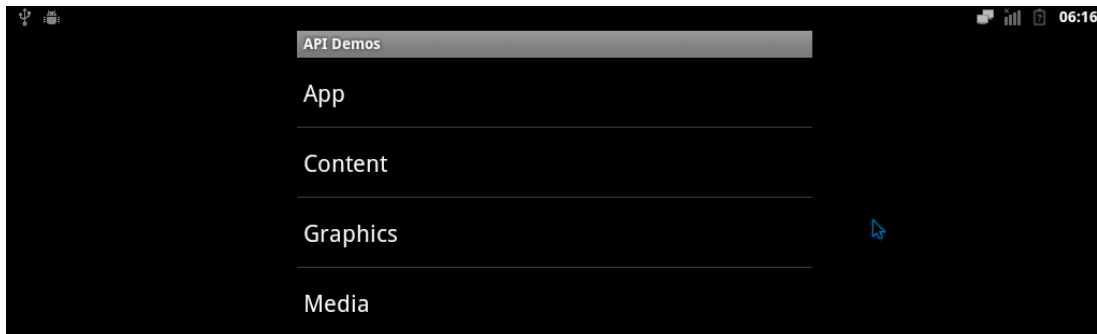
To create an AVD:

1. In the AVD Manager panel, click New to bring up the 'Create new Android Virtual Device (AVD)' dialogue.
2. Typically you will just fill in the "Name" and "Target" fields for basic AVD configuration. The rest fields will be completed automatically with default values when you select the API level to which you want to test against in the "Target" field. The AVD can be further customized if needed.



3. Click Create AVD. Your AVD is now ready.

4. Select the AVD you just created and click Start to launch the emulator.



Now remove your NSD and refer to the getting started software for how the board can be connected to the LCD screen and your computer with the connectors.

Installing the USB Driver

After connecting your NSD to your computer the first time, follow the “Welcome to the hardware update” wizard to install and download the driver for Android Composite ADB Interface.

1. Make sure “Installation from a list or specific location (Advanced)” is checked and Click Next.
2. Choose “Search for the best driver in these locations” and check “Include this location in the search”.
3. Type or browse the path that is equivalent to \Android\android-sdk\extras\google\usb_driver after you specify the exact CD-ROM Drive file where you have your Android installation package located.
4. Click Next to install the driver.
5. Restart your computer for the installation to take effect.

How to Use ADB

From here you can easily use adb to communicate with your device in any way you would like. To ensure if the adb has been set up properly on your computer,

1. Run command prompt
2. Change the current working directory to the path where you put db executable file on \platform-tools directory. This should look something like: C:\Program Files\Android\android-sdk\platform-tools
3. Type ‘adb devices’ and hit enter.

```
E:\Program Files\Android\android-sdk\platform-tools>adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
0123456789ABCDEF      device
```

4. If your device is connected successfully, it will be shown up with a serial number.

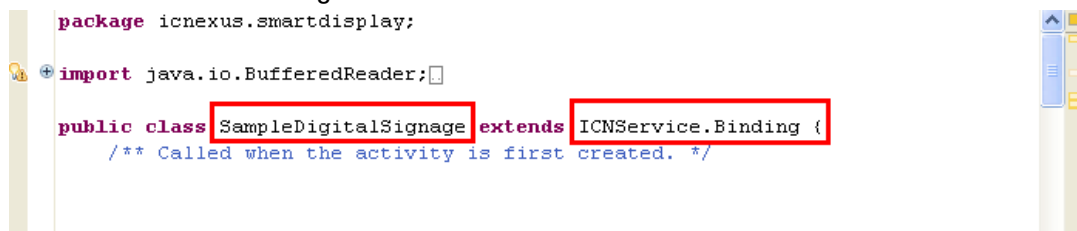
Chapter 3: Guide to Build a Sample Application on NSD

This chapter guides you through various procedures on how you could modify a NSD Sample.

Sample Digital Signage

Here we will introduce how Clock and Ticker can be added to the DigitalSignage project.

1. Declare a public class named SampleDigitalSignage that extends the ICNService.Binding class

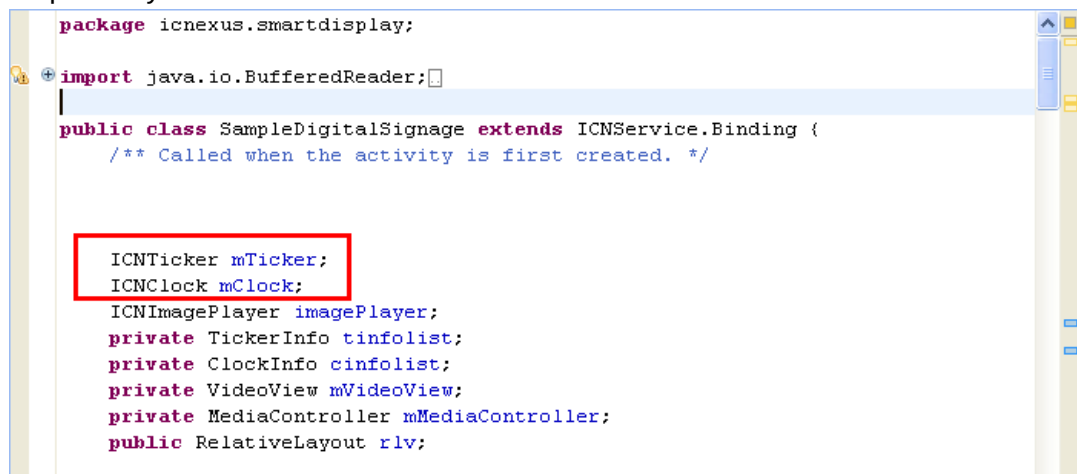


```
package icnexus.smartdisplay;

import java.io.BufferedReader;

public class SampleDigitalSignage extends ICNService.Binding {
    /** Called when the activity is first created. */
```

2. Apply ICNTicker and ICNClock to create two classes named mTicker and mClock respectively.



```
package icnexus.smartdisplay;

import java.io.BufferedReader;

public class SampleDigitalSignage extends ICNService.Binding {
    /** Called when the activity is first created. */

    ICNTicker mTicker;
    ICNClock mClock;
    ICNImagePlayer imagePlayer;
    private TickerInfo tinfoList;
    private ClockInfo cinfoList;
    private VideoView mVideoView;
    private MediaController mMediaController;
    public RelativeLayout rlv;
```

3. Use TickerInfo and ClockInfo to create two parameter objects for Ticker and Clock. Name them as tinfoList and cinfoList respectively.

```

ICNTicker mTicker;
ICNClock mClock;
ICNImagePlayer imagePlayer;
private TickerInfo tinfoList;
private ClockInfo cinfoList;
private VideoView mVideoView;
private MediaController mMediaController;
public RelativeLayout rlv;

```

4. Create a relative layout

```

ICNImagePlayer imagePlayer;
private TickerInfo tinfoList;
private ClockInfo cinfoList;
private VideoView mVideoView;
private MediaController mMediaController;
public RelativeLayout rlv;

/** reading file path*/
private String path;
private String path;

```

5.1 Add a function parameter named TickerSetting for Ticker

```

private int vWidth = 0;
private int vHeight = 0;

private void TickerSetting(){

    int textColor = Color.RED, backgroundColor = Color.TRANSPARENT, textSize;
    int backgroundWidth = 800, backgroundHeight = 150;
    int x = 0, y = 0;
    int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580, ani_end_y = 580;
    int ani_duration = 20000, ani_repeat_time = -1;
    String textData = "Smart Display";

    tinfoList = new TickerInfo(textColor, backgroundColor, textSize,
        backgroundWidth, backgroundHeight,
        x, y,
        ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,
        ani_duration, ani_repeat_time,
        textData);
}

private void TickerDataSetting(String text){
    tinfoList.textData = text;
}

```

5.1.1 Set parameters

```

private void TickerSetting(){
    int textColor = Color.RED, backgroundColor = Color.TRANSPARENT, textSize = 10;
    int backgroundWidth = 800, backgroundHeight = 150;
    int x = 0, y = 0;
    int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580, ani_end_y = 580;
    int ani_duration = 20000, ani_repeat_time = -1;
    String textData = "Smart Display";

    tinfolist = new TickerInfo(textColor, backgroundColor, textSize,
        backgroundWidth, backgroundHeight,
        x, y,
        ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,
        ani_duration, ani_repeat_time,
        textData);
}

```

Parameter definition :

textColor = Ticker's text color

backgroundColor = Ticker's background color

textSize = Ticker's font size

backgroundWidth = the width of Ticker's background

backgroundHeight = the height of Ticker's background

x = Ticker's x coordinate of the upper-left corner

y = Ticker's y coordinate of the upper-left corner

ani_begin_x = Ticker's x coordinate of the starting point at the upper-left corner

ani_end_x = Ticker's x coordinate of the end point at the upper-left corner

ani_begin_y = Ticker's y coordinate of the starting point at the upper-left corner

ani_end_y = Ticker's y coordinate of the end point at the upper-left corner

ani_duration = the time Ticker takes to complete the whole path (ms)

ani_repeat_time = the number of times Ticker has run. -1 implies infinity

textData = the text content of Ticker

5.1.2 Add parameters to tinfoList

```
private void TickerSetting(){  
  
    int textColor = Color.RED, backgroundColor = Color.TRANSPARENT, textSize = 10;  
    int backgroundWidth = 800, backgroundHeight = 150;  
    int x = 0, y = 0;  
    int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580, ani_end_y = 580;  
    int ani_duration = 20000, ani_repeat_time = -1;  
    String textData = "Smart Display";  
  
    tinfoList = new TickerInfo(textColor, backgroundColor, textSize,  
        backgroundWidth, backgroundHeight,  
        x, y,  
        ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,  
        ani_duration, ani_repeat_time,  
        textData);  
}
```

5.2 Add a new parameter for Clock named ClockSetting

```
private void ClockSetting(){  
  
    int window_x = 0, window_y = 615, window_width = 240, window_height = 70;  
    int size = 50, textColor = Color.GREEN, backgroundColor = Color.DKGRAY;  
    Typeface tf = Typeface.MONOSPACE;  
  
    cinfoList = new ClockInfo(window_x, window_y, window_width, window_height,  
        size, textColor, backgroundColor, tf);  
}
```

5.2.1 Set the parameters

```
private void ClockSetting(){  
  
    int window_x = 0, window_y = 615, window_width = 240, window_height = 70;  
    int size = 50, textColor = Color.GREEN, backgroundColor = Color.DKGRAY;  
    Typeface tf = Typeface.MONOSPACE;  
  
    cinfoList = new ClockInfo(window_x, window_y, window_width, window_height,  
        size, textColor, backgroundColor, tf);  
}
```

Parameter definition:

window_x = Clock's x coordinate of the upper left corner of the background

window_y = Clock's y coordinate of the upper left corner of the background

window_width = the width of Clock's background

window_height = the height of Clock's background

size = Clock's font size

textColor = Clock's font color

backgroundColor = Clock's background color

tf = Clock's text font type

5.2.2. Adding parameters to cinfoList

```

private void ClockSetting(){

    int window_x = 0, window_y = 615, window_width = 240, window_height = 70;
    int size = 50, textColor = Color.GREEN, backgroundColor = Color.DKGRAY;
    Typeface tf = Typeface.MONOSPACE;

    cinfoList = new ClockInfo(window_x, window_y, window_width, window_height,
        size, textColor, backgroundColor, tf);
}

```

6. In the onCreate method, call function TickerSetting and ClockSetting

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();

    rlv = new RelativeLayout(this);
    imagePlayer = new ICNImagePlayer(this);
    ImagePlayerSetting();
    mTicker = new ICNTicker(this, tinfoList);

    mClock = new ICNClock(this, cinfoList);
    mVideoView = new VideoView(this);

    //////////////////////////////////////

    mMediaController = new MediaController(this);
    mMediaController.setAnchorView(mVideoView);
}

```

7. In the onCreate method, change the variable of relative layout to the current context.

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();
    rlv = new RelativeLayout(this);
    imagePlayer = new ICNImagePlayer(this);
    ImagePlayerSetting();
    mTicker = new ICNTicker(this, tinfoList);

    mClock = new ICNClock(this, cinfoList);
    mVideoView = new VideoView(this);

    //////////////////////////////////////

    mMediaController = new MediaController(this);
    mMediaController.setAnchorView(mVideoView);
}

```

8. Create two new objects named ICTTicket and ICTClock respectively and then initialize them with the information in their parameters.

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();
    rlv = new RelativeLayout(this);
    imagePlayer = new ICNImagePlayer(this);
    ImagePlayerSetting();
    mTicker = new ICNTicker(this, tinfoList);
    mClock = new ICNClock(this, cinfoList);
    mVideoView = new VideoView(this);

    //////////////////////////////////////

    mMediaController = new MediaController(this);
    mMediaController.setAnchorView(mVideoView);

```

9. In the onCreate method, use rlv (relative layout) object reference to invoke the addView method. Then you apply the mClock and mTicket arguments to the addView method.

```

    rlv.addView(imagePlayer);
    rlv.addView(mClock);
    rlv.addView(mVideoView);
    rlv.addView(mTicker);
    setContentView(rlv);

    init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
    Log.v("SampleDigitalSignage::onCreate", "bindService ...");
    bindService();
}

```

10. In the onCreate method, add setContentView to the relative layout object reference.

```

    rlv.addView(imagePlayer);
    rlv.addView(mClock);
    rlv.addView(mVideoView);
    rlv.addView(mTicker);
    setContentView(rlv);

    init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
    Log.v("SampleDigitalSignage::onCreate", "bindService ...");
    bindService();
}

```

11. In the onCreate method, add rlv(relative layout) to the second parameter and the Command ID ICNService to the first parameter, init.

```

        rlv.addView(imagePlayer);
        rlv.addView(mClock);
        rlv.addView(mVideoView);
        rlv.addView(mTicker);
        setContentView(rlv);

        init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
        Log.v("SampleDigitalSignage::onCreate", "bindService ...");
        bindService();
    }

```

12. Call the `bindService()` function

```

        rlv.addView(imagePlayer);
        rlv.addView(mClock);
        rlv.addView(mVideoView);
        rlv.addView(mTicker);
        setContentView(rlv);

        init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
        Log.v("SampleDigitalSignage::onCreate", "bindService ...");
        bindService();
    }

```

13. In the `SampleDigitalSignage` class, override the `onDestroy()` function.

```

@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleDigitalSignage::onDestroy", "unbindService done");

    super.onDestroy();
}

```

14. In the `onDestroy()` function, add both "`unbindService()`" and "`super.onDestroy()`".

```

@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleDigitalSignage::onDestroy", "unbindService done");

    super.onDestroy();
}

```

SampleChart

This section describes how to add an application to your Chart Project.

1. Declare a new project named 'SampleChart' that extends the ICNService.Binding class.

```
import org.achartengine.GraphicalView;

public class SampleChart extends ICNService.Binding {
    //private IDemoChart[] mCharts = new IDemoChart[] {new SalesGrowthChart()};
    private IDemoChart mCharts = new SalesGrowthChart();
```

2. Use IDemoChart to add a new class object named mCharts. Assign mCharts to SalesGrowthChart()

```
public class SampleChart extends ICNService.Binding {
    //private IDemoChart[] mCharts = new IDemoChart[] {new SalesGrowthChart()};
    private IDemoChart mCharts = new SalesGrowthChart();
```

3. Use ChartInfo to establish a parameter object for Chart named cinfoList.

```
//private String[] mMenuSummary;
private Handler handler = new Handler();

private ChartInfo cinfoList;

private Vector<Double> XAxis = new Vector<Double>();
```

4. Use IBInfo to establish a parameter object for ImageButton named ibi

```
IBInfo ibi;
ImageButton ib;
```

5. Add two Double data types for each x-axis and y-axis respectively.

```
private ChartInfo cinfoList;

private Vector<Double> XAxis = new Vector<Double>();
private Vector<Double> YAxis = new Vector<Double>();
private Vector<Double> DataYAxis = new Vector<Double>();
```

6. Declare the parameters that will be used in ChartInfo.

```

private Vector<Double> YAxie = new Vector<Double>();
private Vector<Double> DataYAxie = new Vector<Double>();

String ChartTitle = "Chart Demo", XName = "X", YName = "Y";
String ChartExplain = "Random value chart diagram";
double XMin = 0, XMax = 20, YMin = 0, YMax = 15;
int ColorAxes = Color.LTGRAY, ColorLabel = Color.YELLOW;
int ColorChart = Color.RED, ColorBackgground = Color.rgb(100, 50, 200);
int XRangeNum = 5, YRangeNum = 5;
boolean ShowGrid = false, BackgroundColorApply = true;
boolean ZoomButtonsVisible = false, ShowLegend = false;
boolean ShowAxes = true, ShowLabels = true;
float ChartSmoothness = 0.33f;

int XAxieNumber = 21;
int YAxieNumber = 21;
int ChartRefreshTime = 500;

```

Parameter definition:

XMin : The minimum Chart's x-axis value

XMax : The maximum Chart's x-axis value

YMin : The minimum Chart's y-axis value

YMax : The maximum Chart's y-axis value

ColorAxes : Chart's axes color

ColorLabel : Chart's label color

ColorChart : Chart's line color

ColorBackgground : Chart's background color

XRangeNum : The number of Chart's x-axis intervals

YRangeNum: The number of Chart's y-axis intervals

ChartSmoothness : Chart's line smoothness

ShowGrid : The Boolean value to specify whether to display or hide gridlines in Chart

BackgroundColorApply : The Boolean value to specify whether the background color is applied to Chart. A false means 'transparent' background and a true will change the ColorBackgground to the background color.

ZoomButtonsVisible: The Boolean value to specify whether the zoom interface is displayed

ShowLegend : The Boolean value to specify whether Chart's text description at the bottom left corner is displayed

ShowAxes: The Boolean value to specify whether Chart's axes are displayed.

ShowLabels: The Boolean value to specify whether the label names of Chart's axes are displayed.

ChartTitle: The Boolean value to specify whether Chart's title name is displayed.

XName : The name of Chart's x-axis.

YName : The name of Chart's y-axis.

ChartExplain : Chart's text description at the bottom left corner

7. Add a GraphicalView object named cChartView

```
int ChartRefreshTime = 500;
```

```
public GraphicalView cChartView;  
public RelativeLayout rlv;  
protected LayoutParams mlp;
```

8. Add a UserImageButton object named ib.

```
IBInfo ibi;  
UserImageButton ib;
```

9. Add a new relative layout

```
int ChartRefreshTime = 500;  
  
public GraphicalView cChartView;  
public RelativeLayout rlv;  
protected LayoutParams mlp;
```

10. Add a LayoutParams named mlp.

```
int ChartRefreshTime = 500;  
  
public GraphicalView cChartView;  
public RelativeLayout rlv;  
protected LayoutParams mlp;
```

11. In the function ChartInfo(), assign all the parameters that have already established in Step 6 to cinfoList.

```
public void info_update()  
{  
    cinfoList = new ChartInfo(XMin, XMax, YMin, YMax, ColorAxes, ColorLabel,  
        ColorChart, ColorBackground, XRangeNum, YRangeNum, ChartSmoothness,  
        ShowGrid, BackgroundColorApply, ZoomButtonsVisible, ShowLegend,  
        ShowAxes, ShowLabels,  
        ChartTitle, XName, YName, ChartExplain);  
}
```

12. For the range of values in X-axis and Y-axis that have already defined in Step 6, add a double data type to XAxe and YAxis respectively.

```
for(int i=0; i<XAxeNumber; i++)  
{  
    XAxe.addElement((double) i);  
}  
for(int j=0; j<YAxeNumber; j++)  
{  
    YAxe.addElement(Math.random() * 10);  
}
```

13. In the function change(), enter the parameter value for myChart.s.

```

//cChartView = mCharts[0].change(this, cinfoList, XAxe, YAxe);
cChartView = mCharts.change(this, cinfoList, XAxe, YAxe);
mIp = new LayoutParams(500, 350);
mIp.setMargins(250, 150, 0, 0);
cChartView.setLayoutParams(mIp);

```

14. Assign a new LayoutParams parameter to mIp. Then assign two values to LayoutParams to represent the width and height of the layout. Use setMargins to position layout where the first two parameters represent the x-axis coordinate and the y-axis coordinate.

Assign the height and width of LayoutParams to mIp and add mIp to setLayoutParams before cChartView.

```

//cChartView = mCharts[0].change(this, cinfoList, XAxe, YAxe);
cChartView = mCharts.change(this, cinfoList, XAxe, YAxe);
mIp = new LayoutParams(500, 350);
mIp.setMargins(250, 150, 0, 0);
cChartView.setLayoutParams(mIp);

```

15. You can add the parameters of the image button to ibi. Below is the parameter definition:

int x : the x-coordinate of the image button
int y : the y-coordinate of the image button
int w : the width of the image button
int h : the height of the image button
String img : to use the image name as the image button
int did : the numerical value in the image button
int textcolor : the text color of the image button
FONT_ID fontid : the font type of the image button
String text : the text content of the image button

```

//////////exit button//////////
ibi = new IInfo(800, 400, 120, 100, "", R.drawable.blue_blank, Color.RED, ICN1);
ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);

```

16. The parameters of UserImageButton are the id serial numbers for Activity and image button. You may use setButtonAuto to decide whether image button will be automatically turned on

```

//////////exit button//////////
ibi = new IInfo(800, 400, 120, 100, "", R.drawable.blue_blank, Color.RED, ICN1);
ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);

```

17. Use `setFontSize` to specify the font size of the image button. `createButton` will create a new image button based on the parameters of `IBInfo`. `setText` will specify the text content of the image button.

```
ib.setFontSize(60);
ib.createButton(ibi);
ib.setText("Exit");
////////////////////////////////////
```

18. In the `onCreate`, put the view objects of `cChartView` and `ib` inside the `addView()` function. Then add view objects to the already established `rlv` (relative layout).

```
rlv.addView(cChartView);
rlv.addView(ib);

setContentView(rlv);
```

19. In the `onCreate`, add `setContentView(rlv)`. `rlv` means the relative layout.

```
rlv.addView(cChartView);
rlv.addView(ib);

setContentView(rlv);
```

20. In the `onCreate`, add `init(parameter1, parameter2)`. Fill in the `rlv` (relative layout) in `parameter2` and the Command ID of `ICNService` in `parameter1`.

```
init(ICNService.ICN_COMMAND_ID.ICN_CHART_DEMO, rlv);
Log.v("SampleChart::onCreate", "bindService ...");
bindService();
```

21. In the `onCreate`, add `"bindService()"`

```
init(ICNService.ICN_COMMAND_ID.ICN_CHART_DEMO, rlv);
Log.v("SampleChart::onCreate", "bindService ...");
bindService();
```

22. In the `SampleDigitalSignage`, override the `onDestroy` function

```
@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleChart::onDestroy", "unbindService done");

    super.onDestroy();
}
```

23. In the `onDestroy` function, add `"unbindService()"` and `"super.onDestroy()"`.

```
@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleChart::onDestroy", "unbindService done");

    super.onDestroy();
}
```

SampleUserDragAndDrop

This section will introduce you how to add applications in the UserDragAndDrop Project.

1. Create a new Project named SampleUserDragAndDropActivity and extends it to ICNService.Binding

```
import icnexus.smartdisplay.*;

public class SampleUserDragAndDropActivity extends ICNService.Binding {
    RelativeLayout rlv;
    final int imgw = 100, imgh = 100;
```

2. Add a new relative layout

```
import icnexus.smartdisplay.*;

public class SampleUserDragAndDropActivity extends ICNService.Binding {
    RelativeLayout rlv;
    final int imgw = 100, imgh = 100;
```

3. Create a class named UserImageButton and extends it to ICNImageButton

```
public class UserImageButton extends ICNImageButton implements ICNImageButtonCF {
    public UserImageButton(Activity act, int bid) {
        super(act, bid);
    }
    public void buttonDownCallback() {
        Log.v("UserImageButton", "buttonDownCallback ...");
    }
    public void buttonUpCallback() {
        Log.v("UserImageButton", "buttonUpCallback ...");
        exit();
    }
}
```

4. In the class UserImageButton, create a UserImageButton function and add a super(). In the class UserImageButton, create a buttonDownCallback function. In the class UserImageButton, add buttonUpCallback function and exit().

```
public class UserImageButton extends ICNImageButton implements ICNImageButtonCF {
    public UserImageButton(Activity act, int bid) {
        super(act, bid);
    }
    public void buttonDownCallback() {
        Log.v("UserImageButton", "buttonDownCallback ...");
    }
    public void buttonUpCallback() {
        Log.v("UserImageButton", "buttonUpCallback ...");
        exit();
    }
}
```

5. Create a new function named: `dragPictureAndDropToTrash`. This will initialize a draggable item, a dragged zone and `ICNDragSurface`.

```
public void dragPictureAndDropToTrash() {  
    ICNDragSurface dragSurface = new ICNDragSurface(this);  
    dragSurface.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));  
    ICNDragAndDropManager.getInstance().init(dragSurface);  
}
```

6. In the `dragPictureAndDropToTrash` function, add a `dragsurface` next to `ICNDragSurface` and give it a property.

```
public void dragPictureAndDropToTrash() {  
    ICNDragSurface dragSurface = new ICNDragSurface(this);  
    dragSurface.setLayoutParams(new LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));  
    ICNDragAndDropManager.getInstance().init(dragSurface);  
}
```

7.1. In the `dragPictureAndDropToTrash` function, add a new trash icon for `ImageView` named `trashV`. Remember to give it a location, height and width values.

```
ImageView trashV = new ImageView(this);  
trashV.setBackgroundResource(R.drawable.empty_trash);  
LayoutParams lpv = new LayoutParams(128, 200);  
lpv.setMargins(100, 300, 0, 0);  
trashV.setLayoutParams(lpv);  
ICNDropZone dropzone1 = new ICNDropZone(trashV, dropzoneListener);  
ICNDragAndDropManager.getInstance().addDropZone(dropzone1);
```

7.2. In the `dragPictureAndDropToTrash` function, add an `ICNDrop Area` named `dropzone1`. Make the region of `trashV` as the place where one can drop an item

```
ImageView trashV = new ImageView(this);  
trashV.setBackgroundResource(R.drawable.empty_trash);  
LayoutParams lpv = new LayoutParams(128, 200);  
lpv.setMargins(100, 300, 0, 0);  
trashV.setLayoutParams(lpv);  
ICNDropZone dropzone1 = new ICNDropZone(trashV, dropzoneListener);  
ICNDragAndDropManager.getInstance().addDropZone(dropzone1);
```

8.1. In the `dragPictureAndDropToTrash` function, add a draggable object: `ImageView` and named it `i1`. Specify the source of the image, its height, width and position.

```
ImageView i1 = new ImageView(this);  
i1.setBackgroundResource(R.drawable.android_128);  
LayoutParams lp1 = new LayoutParams(imgw, imgh);  
lp1.setMargins(300, 100, 0, 0);  
i1.setLayoutParams(lp1);  
  
i1.setOnTouchListener(onTouchListener);
```

8.2. In the dragPictureAndDropToTrash function, add a draggable object i1 to TouchListener.

```
ImageView i1 = new ImageView(this);
i1.setBackgroundResource(R.drawable.android_128);
LayoutParams lp1 = new LayoutParams(imgw, imgh);
lp1.setMargins(300, 100, 0, 0);
i1.setLayoutParams(lp1);

i1.setOnTouchListener(onTouchListener);
```

9. In the dragPictureAndDropToTrash function, add an exit object: UserImageButton named ib. See below for the parameter definition.

int x : the x-coordinate of the image button

int y : the y-coordinate of the image button

int w : the width of the image button

int h : the height of the image button

String img : To use the image name as the image button

int did : the numerical value in the image button

int textcolor : the text color of the image button

FONT_ID fontid : the font type of the image button

String text : the text content of the image button

Set the ButtonUpAuto as true.

```
IBInfo ibi = new IBInfo(400, 400, 120, 100, "", R.drawable.blue_blank, Color
UserImageButton ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);

ib.setFontSize(60);
ib.createButton(ibi);
ib.setText("Exit");
```

10. In the dragPictureAndDropToTrash function, add a few view objects named: trashV, i1, dragsurface and ib in the addView function to the already established rlv (relative layout).

```
rlv.addView(trashV);
rlv.addView(i1);
rlv.addView(dragSurface);
rlv.addView(ib);
```

```
setContentView(rlv);
}
```

11. In the dragPictureAndDropToTrash function, add setContentView(rlv)

```

        rlv.addView(trashV);
        rlv.addView(i1);
        rlv.addView(dragSurface);
        rlv.addView(ib);

        setContentView(rlv);
    }

```

12. Add onTouchListener for 8.2 to use:

```

    private OnTouchListener onTouchListener = new OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN)
            {

```

13. In the onTouchListener, implement a draggedItem

```

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN)
            {
                ICNDraggableItem draggedItem = new ICNDraggableItem(v,
                    ICNDraggableViewsFactory.getImage(imgw, imgh, R.drawable.a2);

                ICNDragAndDropManager.getInstance().startDragging(this, draggedItem);

                return true;
            }
            return false;
        }
    }

```

14. Add a new event listener named dropzoneListener for 7.2 to use.

```

    private ICNDropZoneEventsListener dropzoneListener = new ICNDropZoneEventsListener() {
        private byte[] userdata = new byte[] { (byte)0x0f, (byte)0x0e, (byte)0x0d, (byte)0x0c, (byte)0x0b, (byte)0x0a, (byte)0x09, (byte)0x08, (byte)0x07, (byte)0x06, (byte)0x05, (byte)0x04, (byte)0x03, (byte)0x02, (byte)0x01, (byte)0x00 };

        @Override
        public void OnDragZoneEntered(ICNDropZone zone, ICNDraggableItem item) {

```

15. In the dropzoneListener, implement OnDragZoneEntered

```

        @Override
        public void OnDragZoneEntered(ICNDropZone zone, ICNDraggableItem item) {
            //zone.getView().setBackgroundColor(Color.YELLOW);
        }
    }

```

16. In the dropzoneListener, implement OnDragZoneLeft

```

@Override
public void OnDragZoneLeft(ICNDropZone zone, ICNDraggableItem item) {
    //zone.getView().setBackgroundColor (Color.parseColor("#999999"));
}

```

17. In the dropzoneListener, implement OnDropped

```

@Override
public void OnDropped(ICNDropZone zone, ICNDraggableItem item) {
    //zone.getView().setBackgroundColor (Color.parseColor("#999999"));
    Toast.makeText(SampleUserDragAndDropActivity.this, "Dropped", 1000).show();
    //rlv.removeAllViews();
    rlv.removeViewAt(1);

    String className = new String("icnexus.smartdisplay.SampleUserDragAndDrop");
    sendUserData(className, userdata, 8);
}

```

18. In the onCreate add rlv = new RelativeLayout(this)

```

//setContentView(R.layout.draganddrop);
rlv = new RelativeLayout(this);
//setContentView(rlv);

```

19. In the onCreate, call the dragPictureAndDropToTrash function that has already established in step 5.

```

dragPictureAndDropToTrash();

```

20. In the onCreate, add "init(parameter1, parameter2)". Fill the rlv (relative layout) in parameter2 and the Command ID of ICNService in parameter 1. Make the demo app as ICN_DRAG_AND_DROP_DEMO

```

init(ICNService.ICN_COMMAND_ID.ICN_DRAG_AND_DROP_DEMO, rlv);

```

21. In the onCreate, add "bindService()"

```

Log.v("SampleUserDragAndDropActivity::onCreate", "bindService ...");
bindService();
try{

```

22. In the SampleUserDragAndDropActivity, override onDestroy function.

```

@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleUserDragAndDropActivity::onDestroy", "unbindService done");

    super.onDestroy();
}

```

23. In the onDestroy function, add "unbindService()" and "super.onDestroy()"

```

@Override
protected void onDestroy() {
    // TODO Auto-generated method stub
    unbindService();
    Log.v("SampleUserDragAndDropActivity::onDestroy", "unbindService done");

    super.onDestroy();
}

```

Chapter 4: Example Applications on NSD

Inputs from the already existing source codes

Users have the ability to load external source code files that have been obtained from other websites or through the developers into Eclipse as a project. The external source codes can be compiled into a specific development platform.

Note1: The development platform could be either a virtual device or a printed circuited board (PCB).

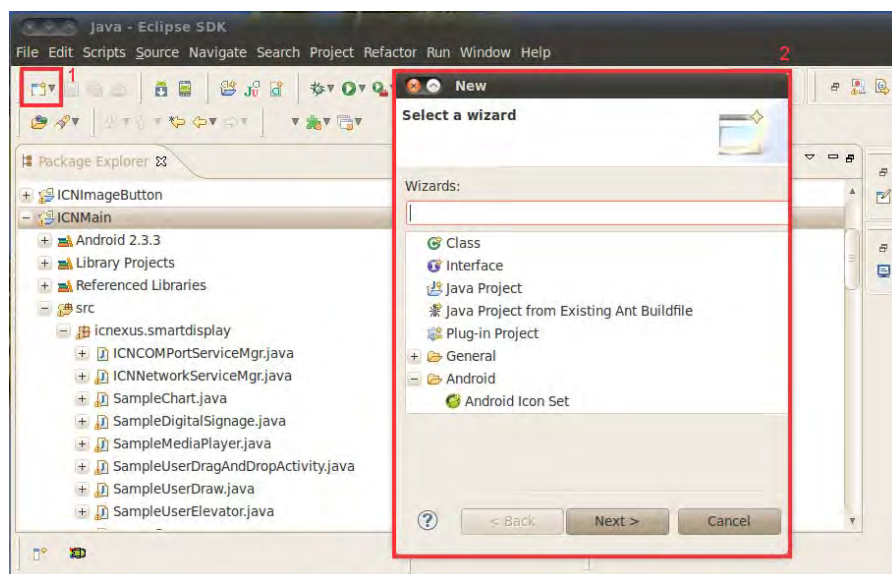
When you have your external source code files and Java Archive files (jar) ready, open Eclipse.

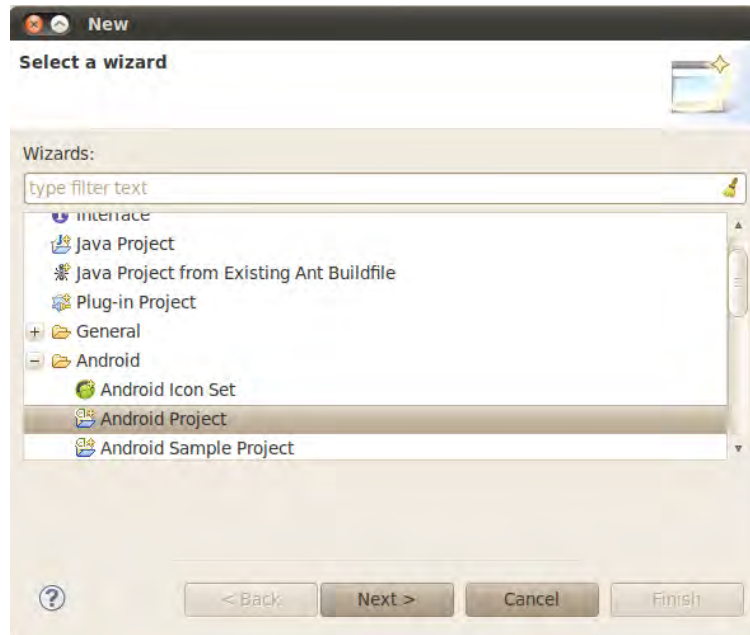
(The system setting below only applies to those software programs that this company has developed. If the files you obtained are from other resources, please modify the setting based on your preferences).

Importing list of source files into an Eclipse Project

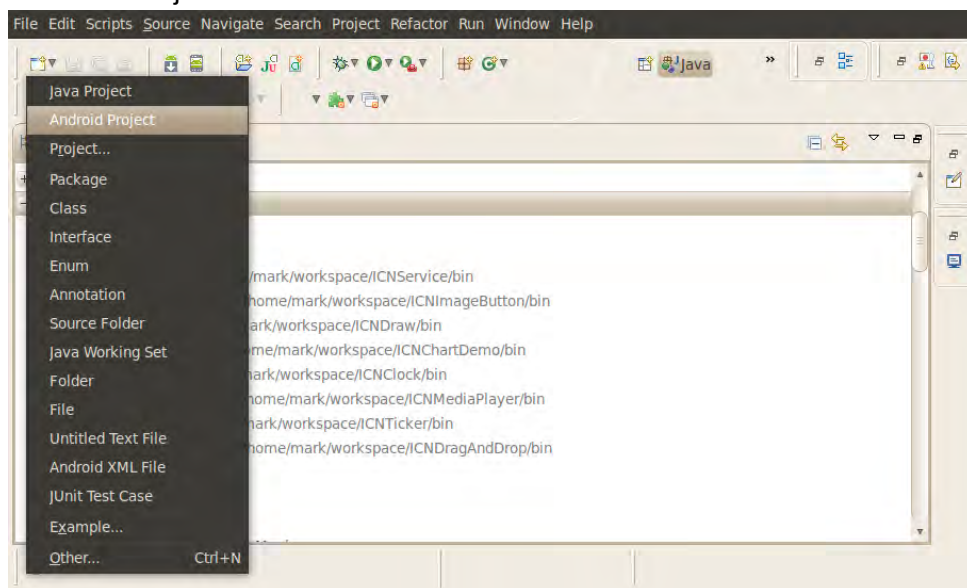
First of all you need to open an Android Project in Eclipse. Below introduces three methods you can use depending on your preference:

- a. Click New at the top left of the screen to bring up the New Window in your Eclipse. Open the Android folder to select Android Project. Click Next.

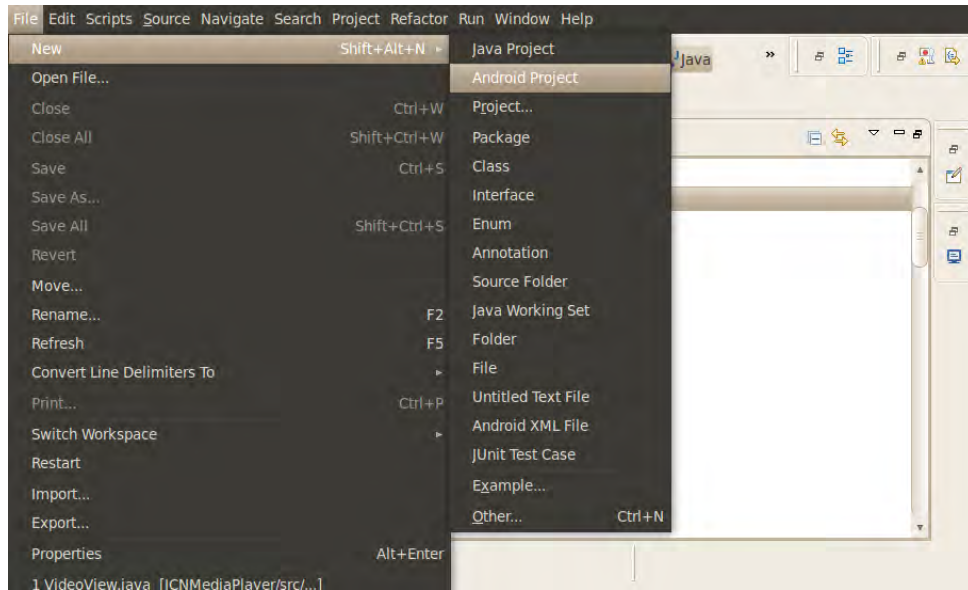




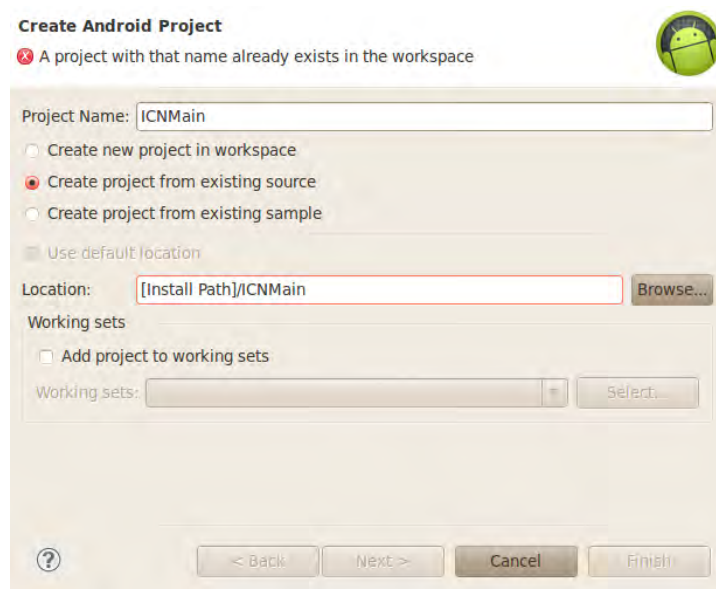
- b. Click the right triangle to the left of the “New” button to bring up a list of resource items to select “Android Project”.



- c. Go to “File” at the menu bar → Choose “New” or Press “Shift+Alt+N” → Select “Android Project”



- d. Select “Create project from existing source” and input “ICNMain” as the Project Name. You can change the project name afterwards.

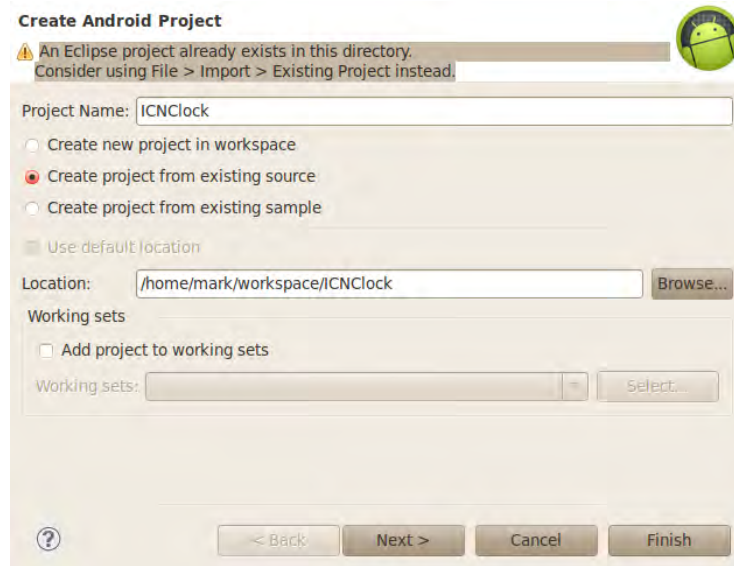


- e. Then, enter the path name of your project directory into the textbox next to the Location field or click the “Browse” button to search your project’s file location. The picture above shows a sample of how the project file ‘ICNMain’ is created.

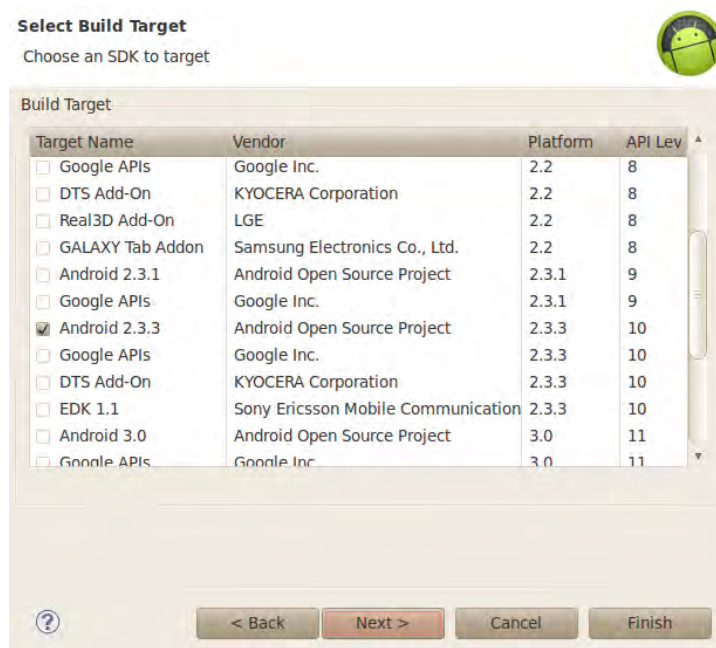
Note: If a dialog “An Eclipse project already exists in this directory” appears at the top of the

window, consider using File>Import>Existing Project instead.

Please continue to follow the steps described in this manual. There is no need to operate the system in accordance with other additional specifications offered by external sources.



- f. After following the previous instructions above, press “Next” to select “Android 2.3.3” as your Build Target. Please check only one SDK version for setup at a time. Never choose more than one version. Click “Next”.

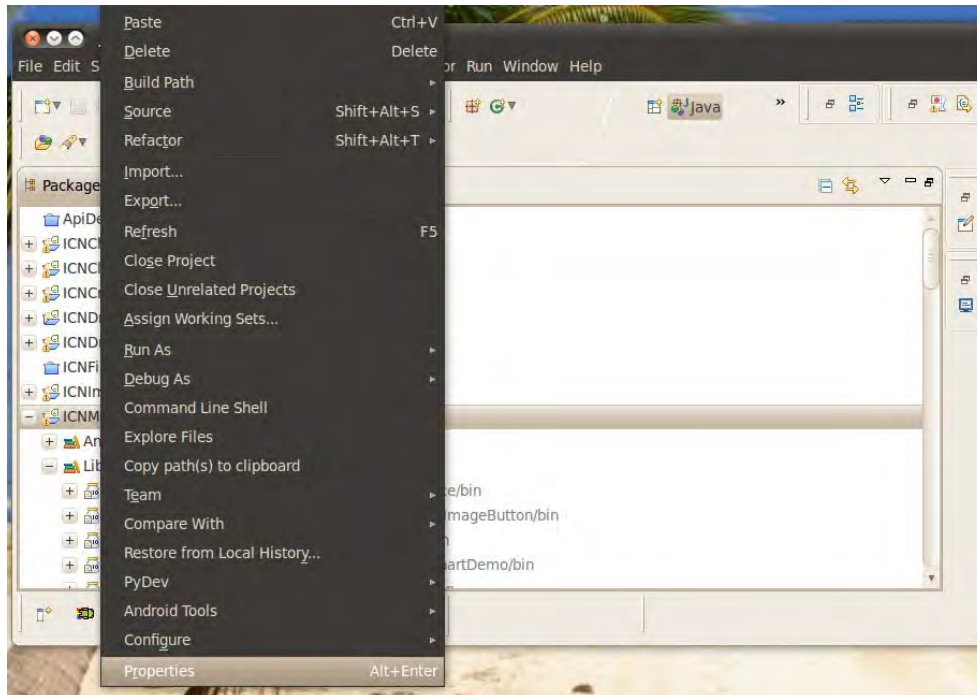


- g. Next, enter your application information to finish the configuration. Input “icnexus.smartdisplay” next to the Package Name field and set the minimum SDK to 10. Click Finish.

At this point a warning message may appear indicating that you have not yet complied the Java files (Java Archive; jar) into the platform.

Adding Extended JAR files to your project

- a. In the Eclipse Package Explore view, right-click the ICNMain folder to open the properties dialog box.

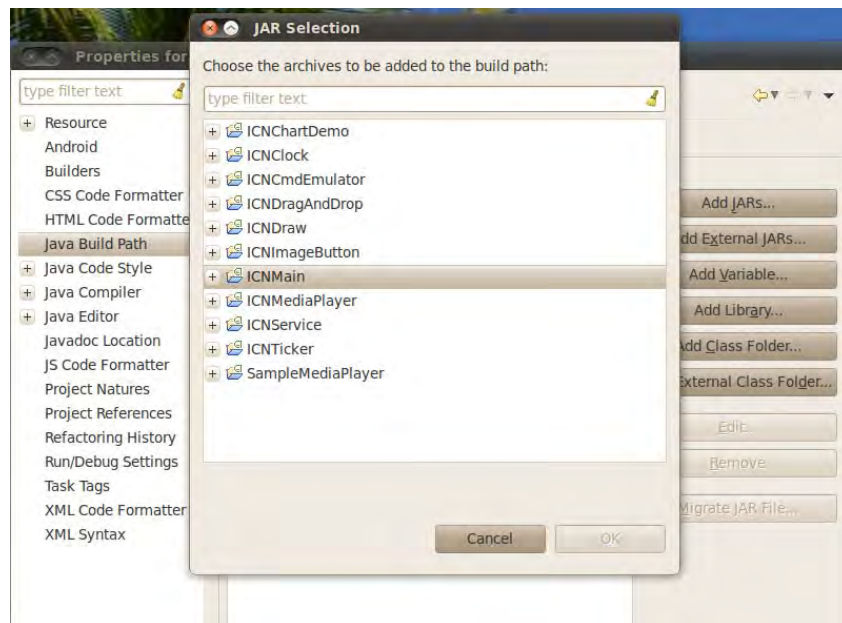


Select “Java Build Path” from the context menu on the left and click the “Libraries” tab.

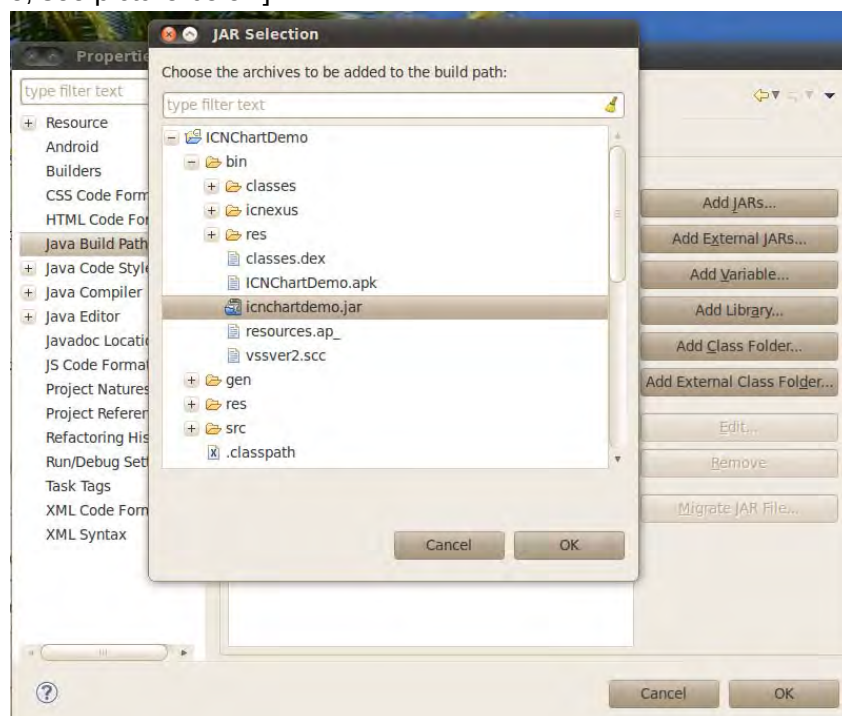


This image is for reference only.

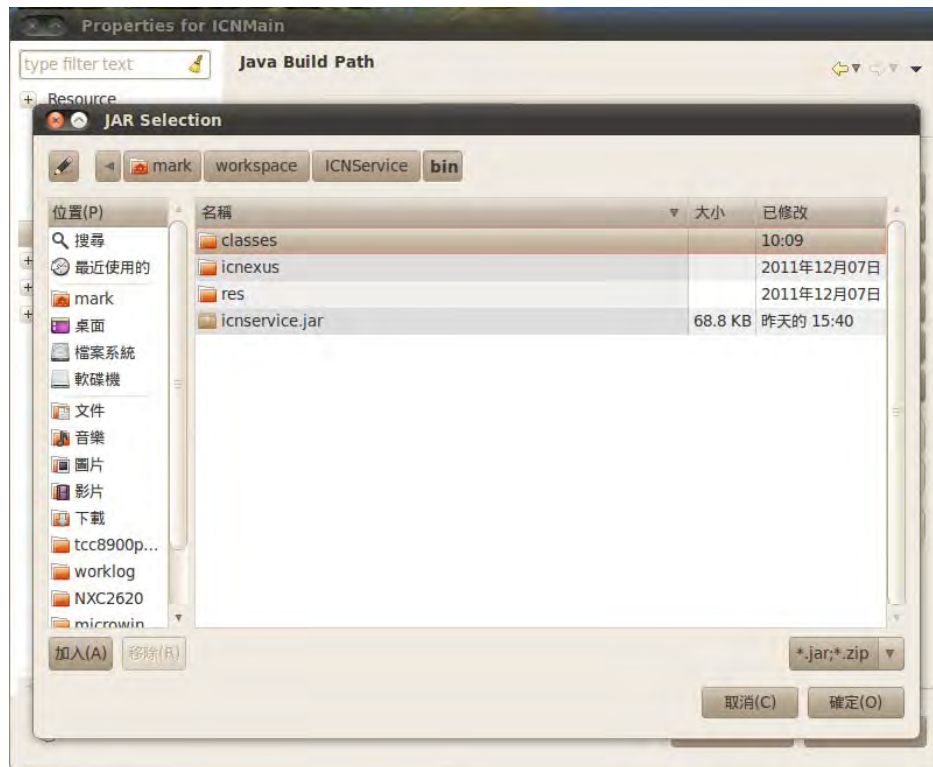
- b. If you would like to add a JAR file to an already existing Java library in Eclipse, press “Add JARs..”.



Under most situations, JAR files will appear inside the bin folder. [Set ICNChartDemo for example; see picture below].



- c. Press “Add External JARs...” if you only want to import an external jar file. You do not have to load the complete project into Eclipse. Instead, you select a JAR file using the file browser.



Adding JAR files from an open project

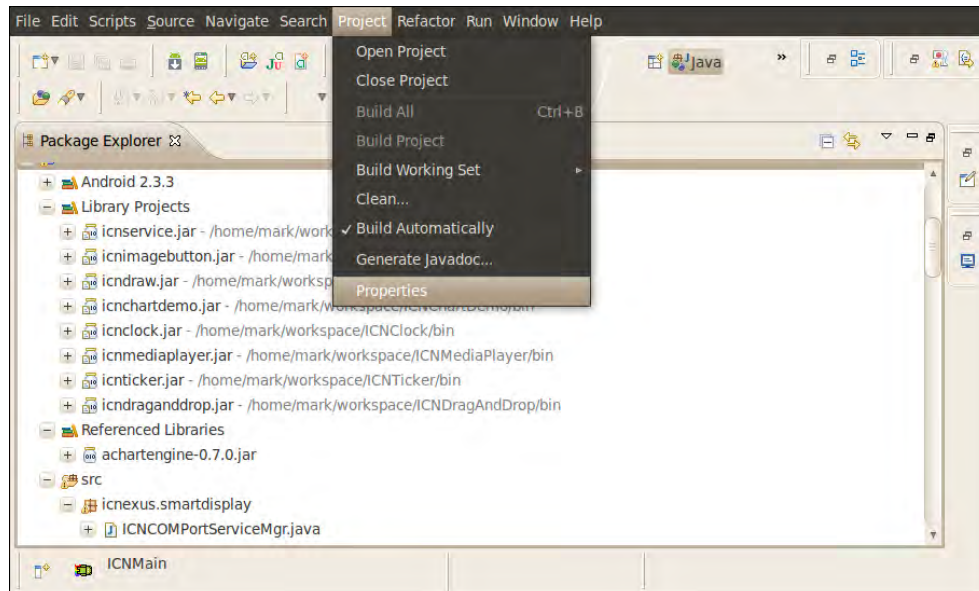
This subsection will describe what steps you should take to develop your own or modify classes. This section will divide into two parts:

- Reference external files In-library project
- Attaching a library from an external project

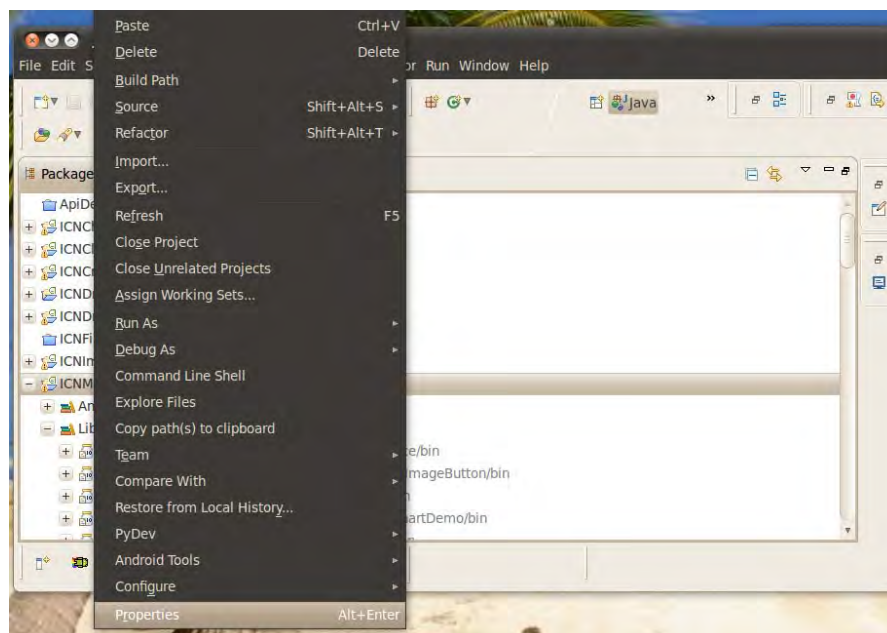
As of December 2011, the new version of Eclipse 3.7.1 has provided more simplicity and efficiency to its setup procedures, implementing all the functionality into the properties of the project.

Please follow the steps below to enter the setup menu:

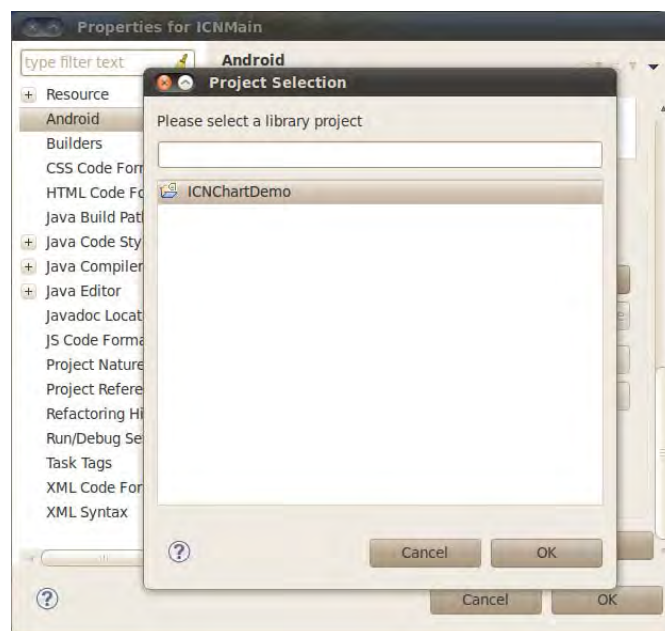
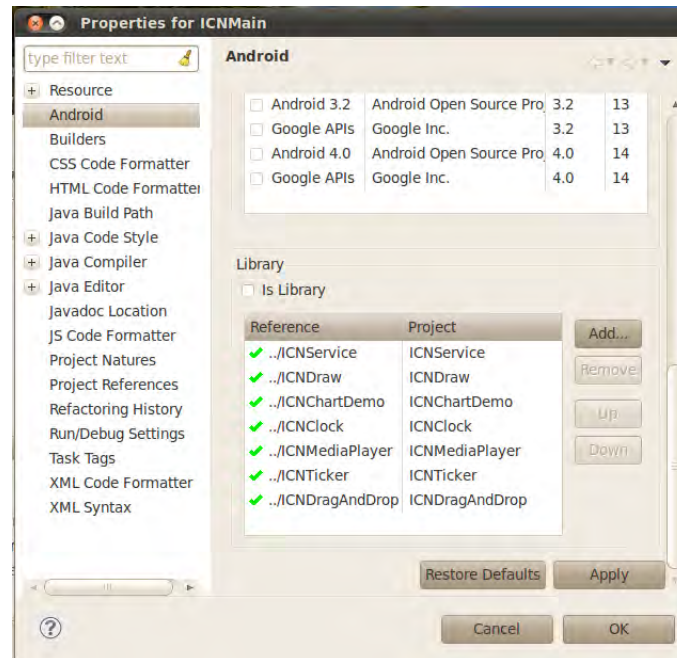
- a. Choose the project name that you would like to set in the Package Explorer view and then click “Project” from the top-menu bar → select “Properties”.



Alternatively, you could also right-click the project name you want to set and then select “Properties”.



- b. When you open the "Properties" window, click the Android option from the context menu on the left.



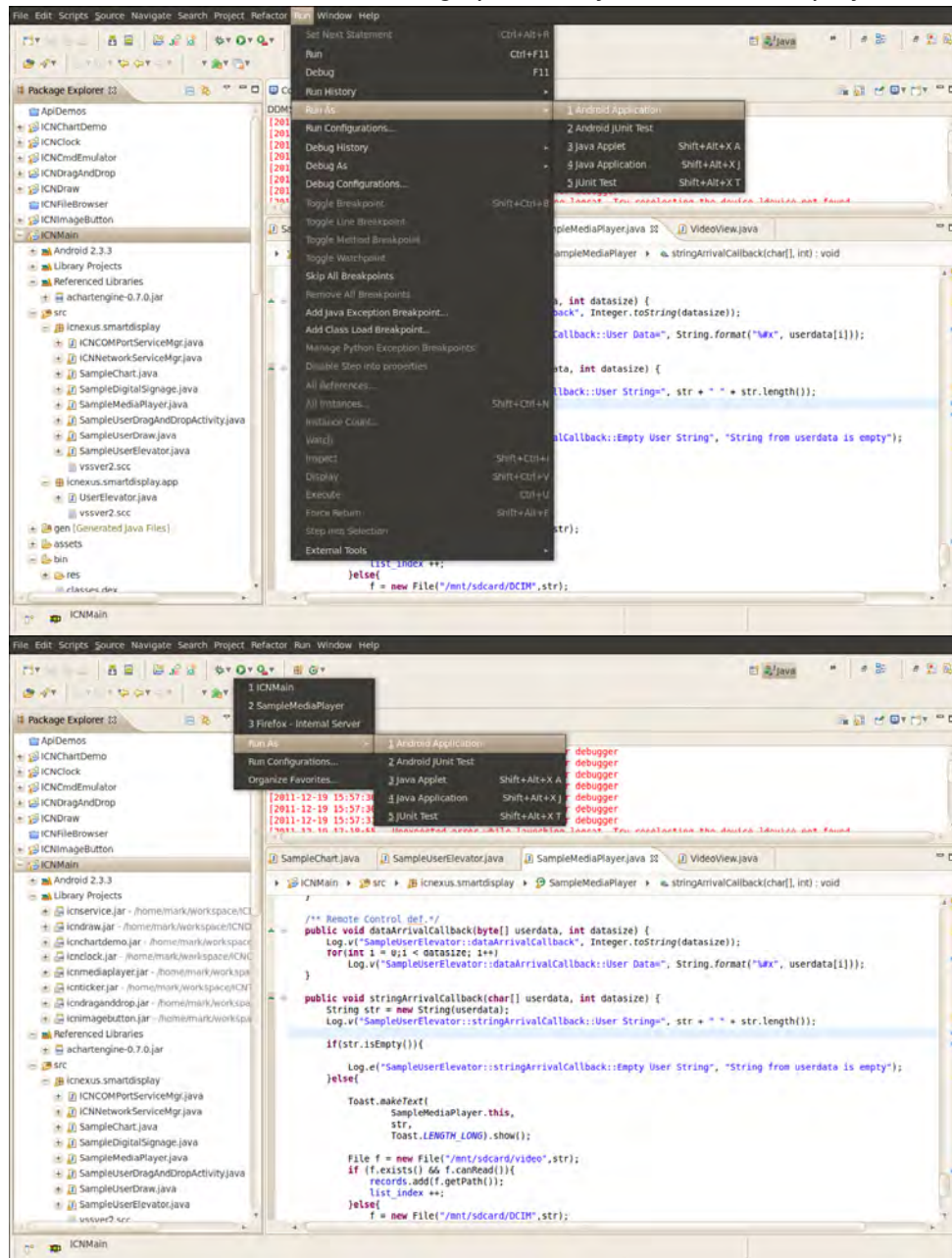
Run the Application

Please ensure that your development platform is already connected to your workstation and this can be normally done via the USB cable.

Select the project name you want to edit, in this case, ICNMain. Then go to Run at the menu bar → Run as → Select Android Application.

Alternatively, you can also press the Run button at the toolbar to select Android Application to run the ICNMain project.

If this is not the first time you run the ICNMain project, click the right triangle to the left of the Run button at the menu bar and this will bring up a list of your most recent project names.



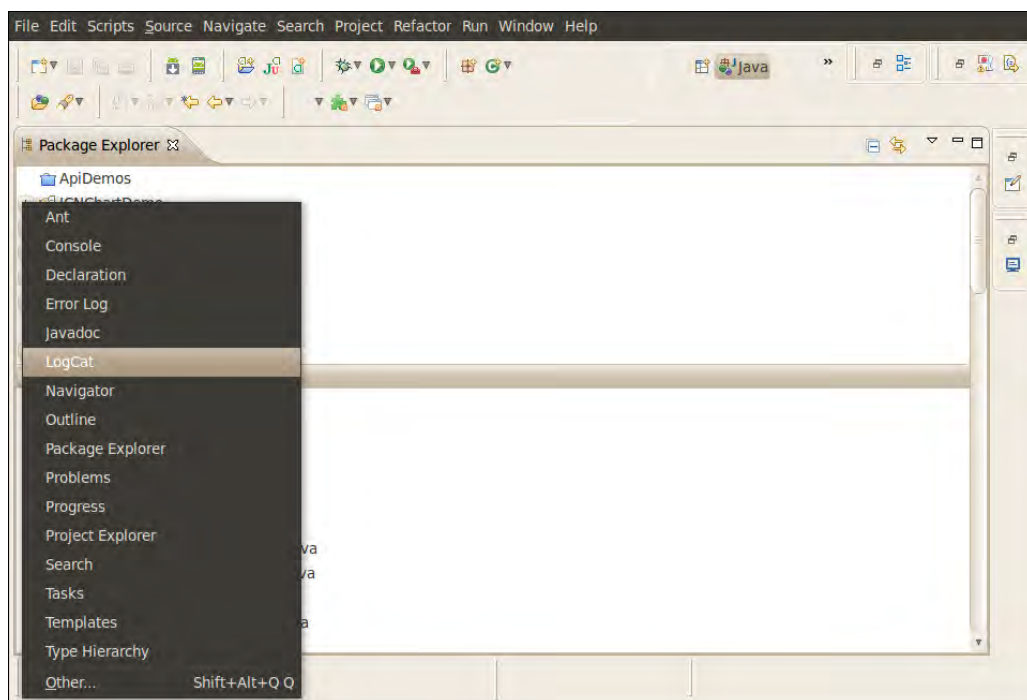
Once this is done, the Eclipse plug-in will automatically upload the APK file from the bin folder in your project package for a new run configuration and then launches the Android emulator. You can now interact with your emulated Android device just as you would do with an actual device.

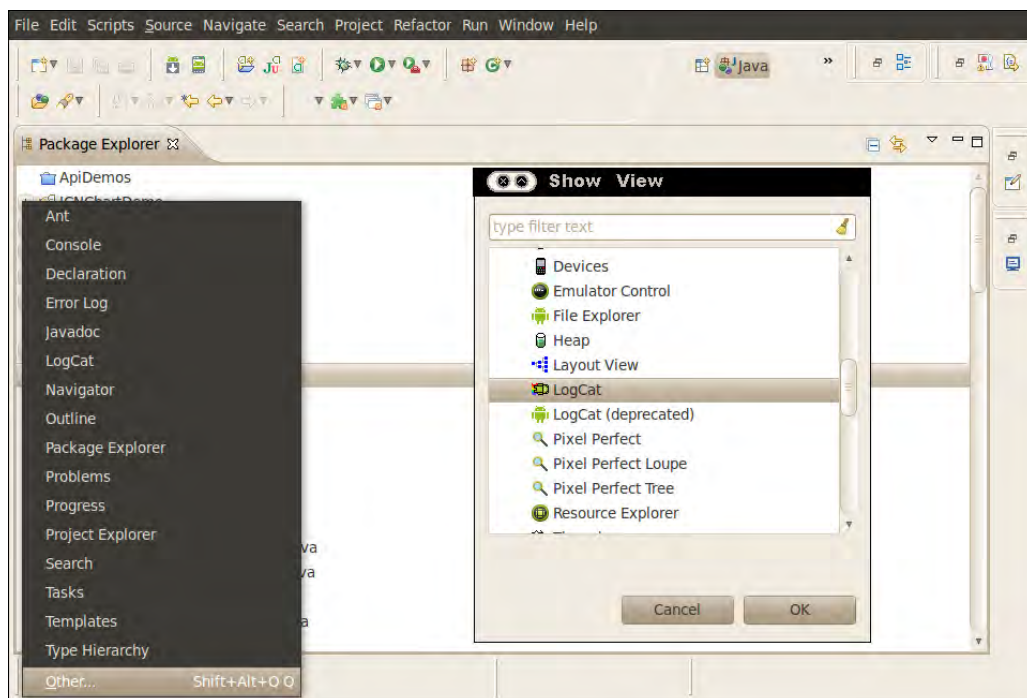
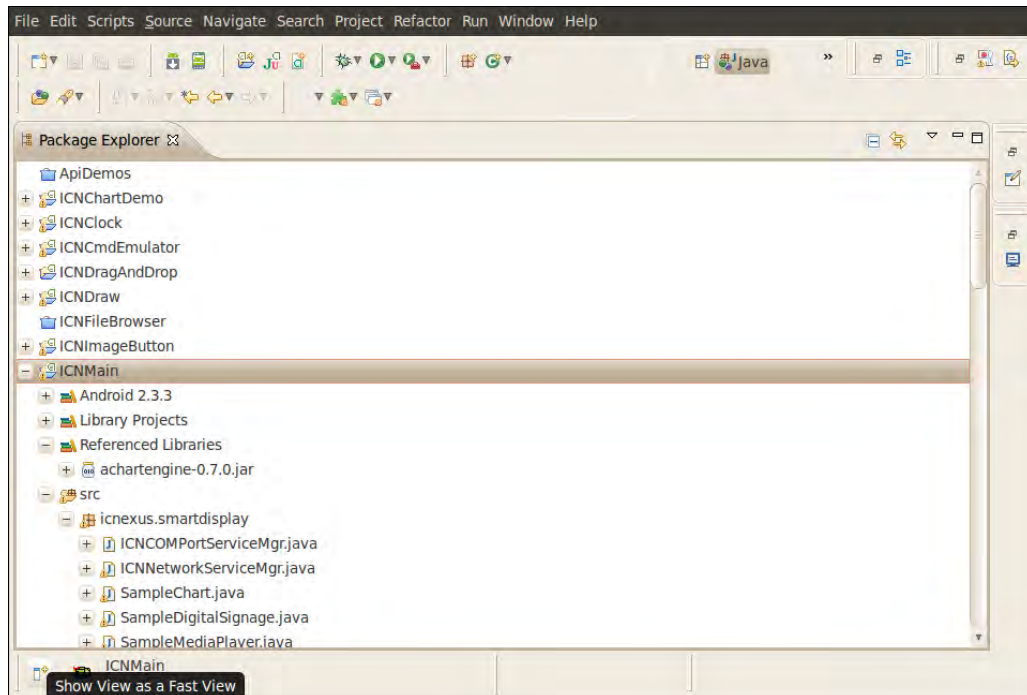
Debugging in Android using Eclipse

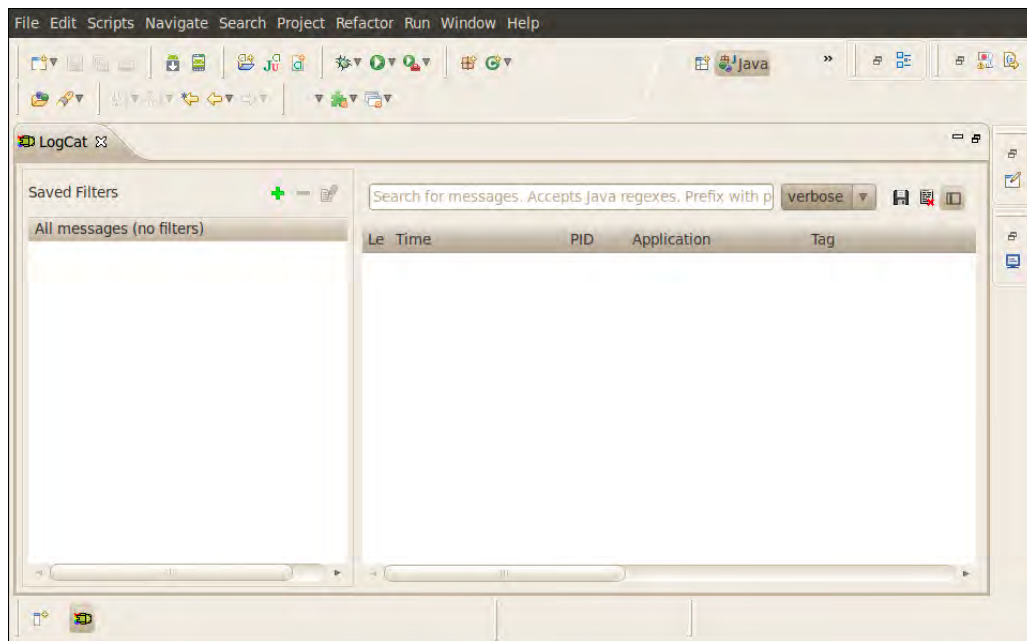
Despite Android being one of the easiest platforms to develop applications for mobile devices, it is not without its flaws. As you are bound to face bugs when creating your applications, you need a good debugging tool. We will use the Android Debug Bridge (ADB) to run on both your android emulator and your development computer to debug a problem with the output of the *adb logcat* command on Eclipse.

To enable logcat in Eclipse for Android, press the button “Show View as a Fast View” at the bottom left of the main Eclipse window and select Logcat.

But if you did not see the Logcat option on the list, then go to Others...→ Android → Logcat.







Chapter 5: Remote Control

How to Run Commands on the NSD Smart Display

Outline :

Task preparation :

Hardware :

Connect your Android device to your computer's USB port

Connect your Android device to an external USB device using RS232 device.

Software :

Install the NSD Smart Display on your Android device.

Install a serial port terminal software that has the capability to send hexadecimal messages to your computer

The introduction of the Smart Display commands

Task preparation

Connect your Android device to your computer's USB port

Connect your Android device to an external USB device using RS232 device.

Install the NSD smart display App on your Android device

Please follow the steps below to import an entire Eclipse project from an archive file. This archive file includes various NSD smart display applications, such as ICNMain, ICNService, ICNChartDemo, ICNClock and ICNDraw.

If there are no error messages from Eclipse, you can now select ICNMain. Press the 'Run' button to install the smart display app on your Android device.

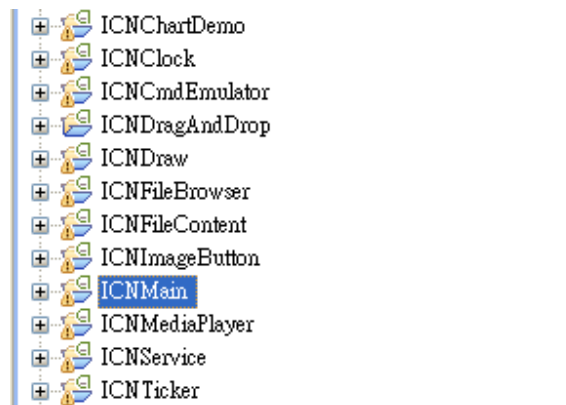


Figure 5-1

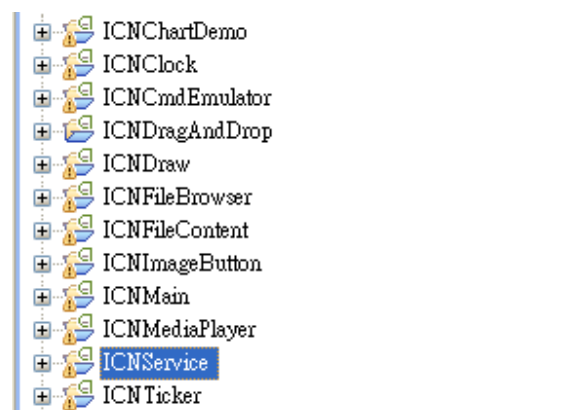


Figure 5-2



Figure 5-3



Figure 5-4

Wait until your device displays a screen image like the one in figure 5-5 to finish the installation process.



Figure 5-5

At this point you can press the “Start Com Port Service” button to activate the Com Port Service. Please note that you must establish this service activation before you are allowed to run the commands to send and receive requests over the network.

If you would to recall the screen image in figure 5.5, please select “Com Port Service Controller” icon from the Menu.

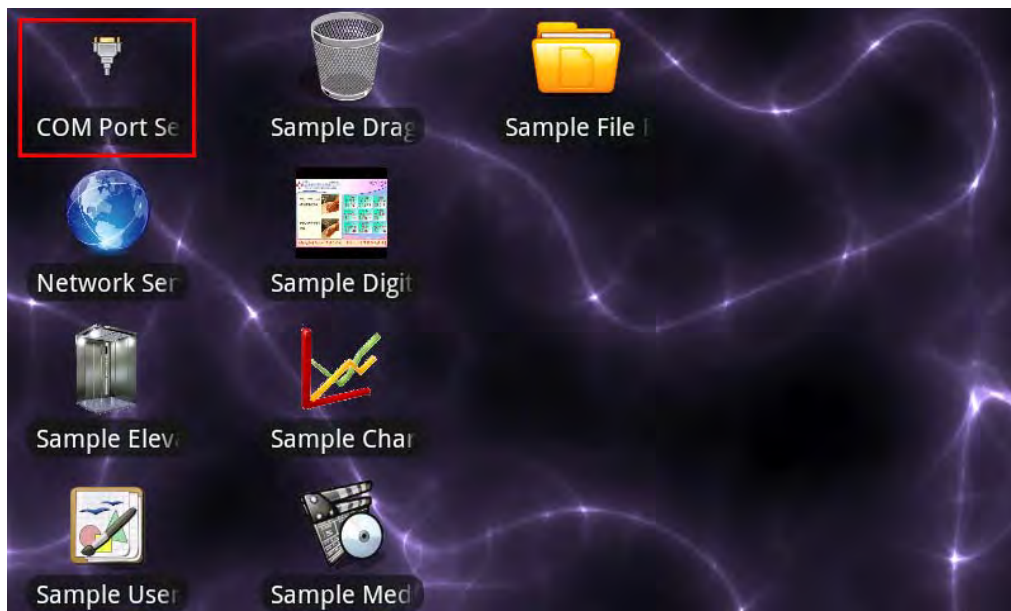


Figure 5-6

After the Com Port Service is activated, you will see an icon imitating the COM Port Service Controller button at the upper left corner.

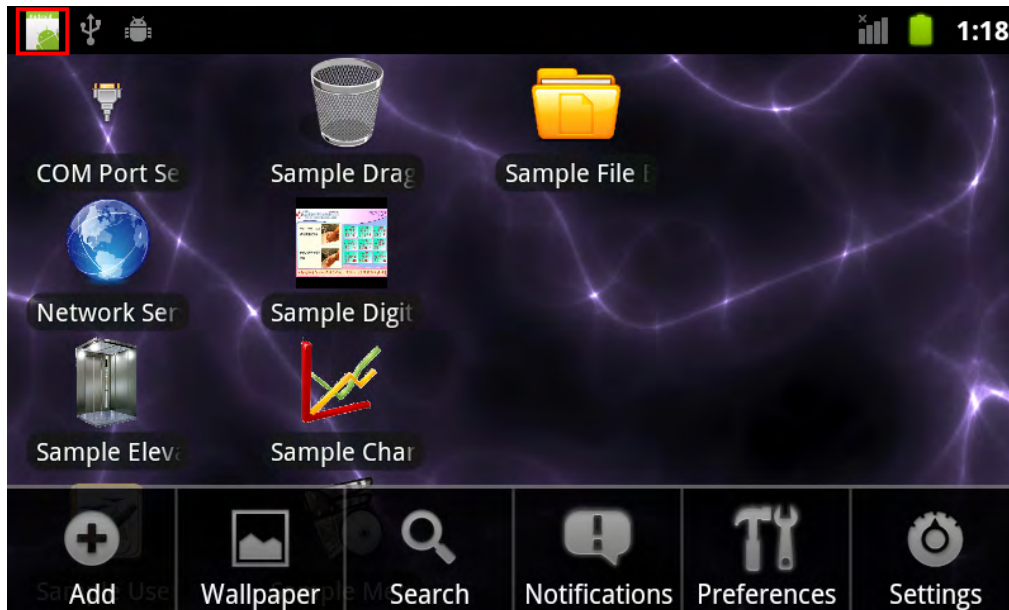


Figure 5-7

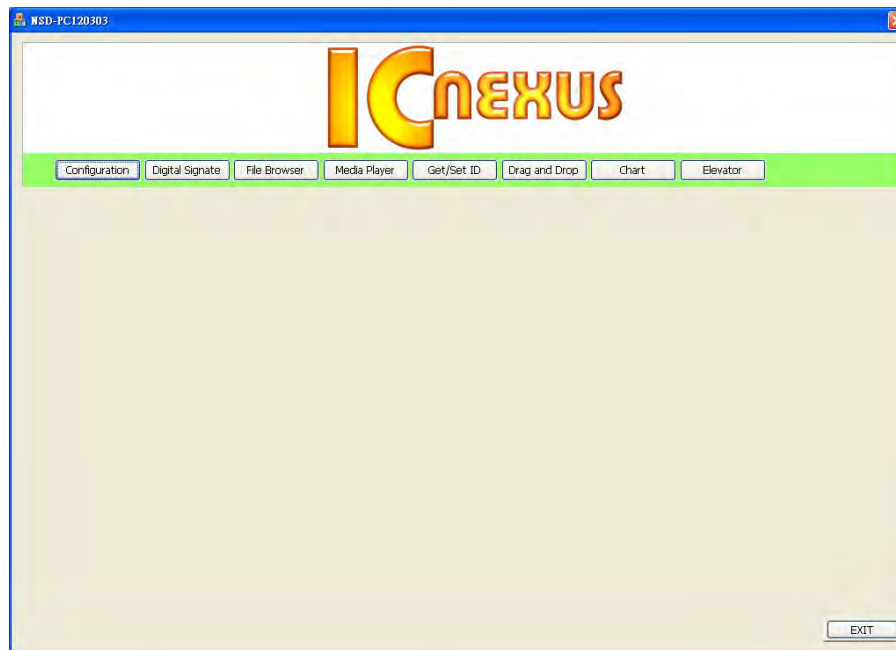


Figure 5-8

If you would like to stop the Com port Service, please press the “Stop COM Port Service” button.

Serial port terminal software installation

(1) Run NSD-PC120303.exe in PC



(2) Open the device manager tab page from Start up -> Set -> Control center -> System

(3) Browse through the device list and locate the (COM and LPT) item

(4) Select the com port that you want to connect the NSD device

For example:

(5). Configure the com port into:

115200 Bits/Sec

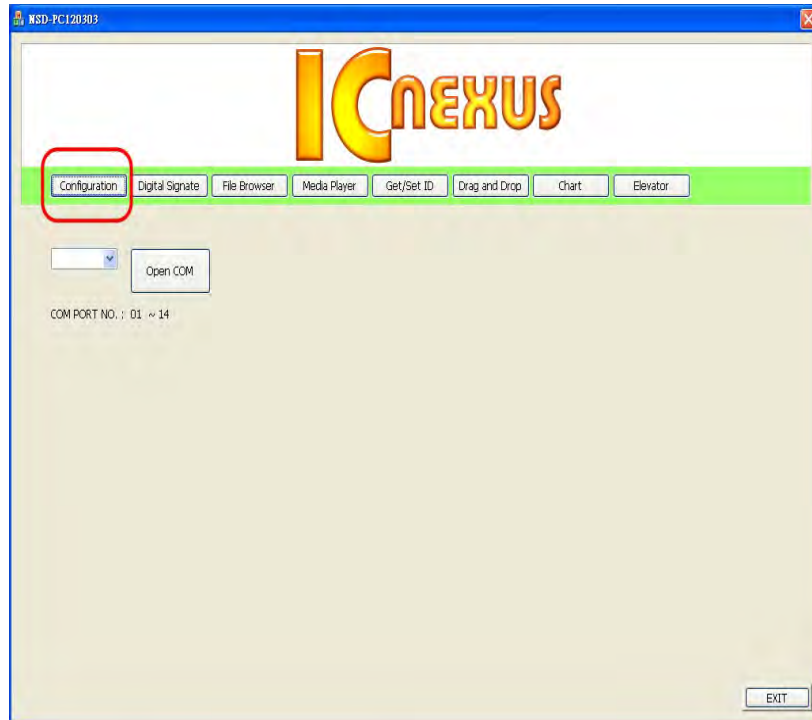
8 Bits/Data

No Parity check

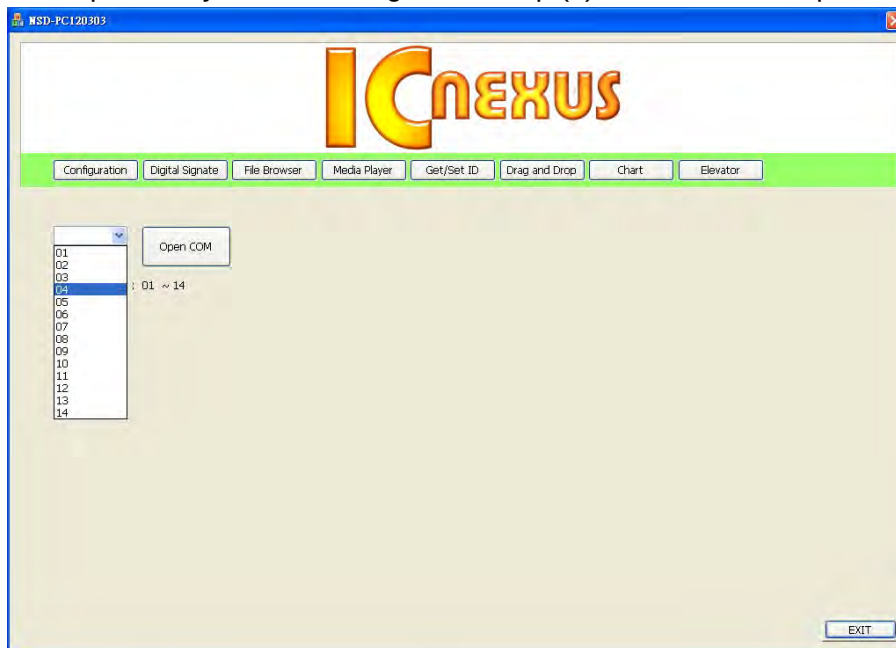
1 Stop Bit

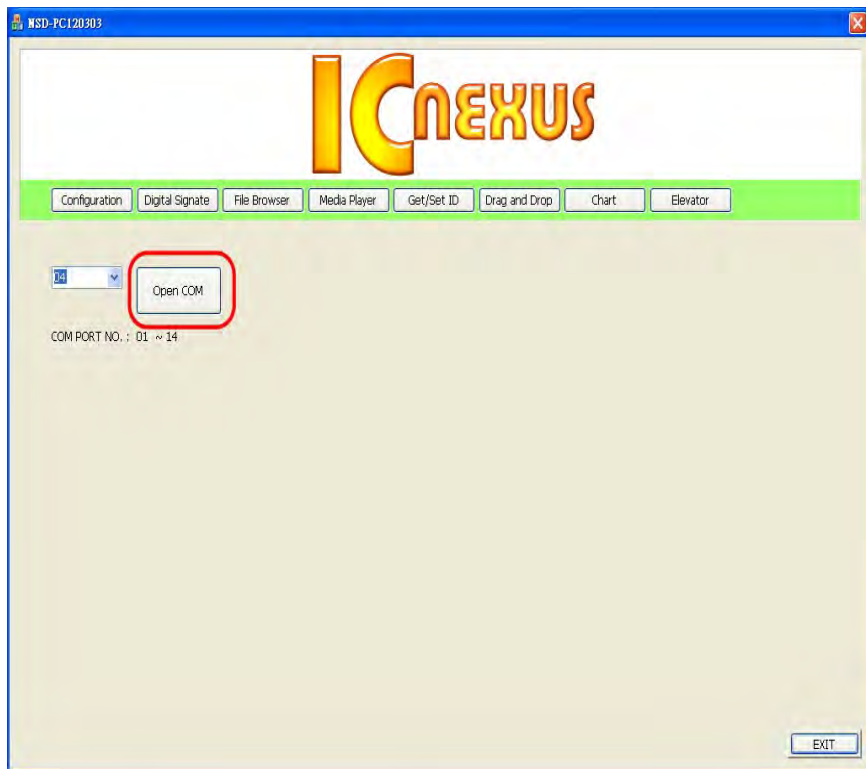
No Flow control

(6) Click "Configuration" button from the dialog NSD-PC120303.exe



(7) Select the com port that you have configured in step (5), then click the “Open COM” button





Appendix A: NSD Software SDK/API

Elevator

The ICNElevator Java class is a simple elevator “activity” in the Android application. It puts you to a role of manager so that you can manage the ICNImageButton class you create. You can extend your code from the ICNElevator Java class and then rewrite your code using the `InitUserButtonSampleCode()` function which can be founded in the the `SampleUserElevator.java` file from the ICNMain project.

- The library projects you will need are ICNService and ICNImageButtonMgr
- `InitUserButtonSampleCode` Function:

```

protected void InitUserButtonSampleCode() {

    rlrv = new RelativeLayout(this);
    int count = 4;
    IBInfo[] ibinfolist = new IBInfo[count];

    DisplayMetrics dm = getScreenSize();

    final int button_width = 500;
    final int button_height = 300;
    int xpos = 0;
    int ypos = -1 * button_height;
    for (int i = 1; i <= count ; i++){
        ypos += button_height;

        if (ypos > dm.heightPixels || ypos + button_height > dm.heightPixels)
        {
            ypos = 0;
            xpos += button_width;
        }

        if (xpos > dm.widthPixels || xpos + button_width > dm.widthPixels)
            break;

        addButton(new ICNImageButton(this, i));
        ibinfolist[i - 1] = new IBInfo(xpos, ypos, button_width, button_height, "", R.drawable.bule_square_button,
            Color.BLUE,
            ICNImageButton.FONT_ID.SANSUS, "");
        getButton(i - 1).setClickedImageId(R.drawable.red_square_button);
        getButton(i - 1).setFontSize( 160 );
        getButton(i - 1).createButton(ibinfolist[i - 1]);

        rlrv.addView(getButton(i - 1));
    }
    init(ICNService.ICN_COMMAND_ID.ICN_ELEVATOR_DEMO, rlrv);

    setContentView(rlrv);
}

```

1. DisplayMetrics ICNService::getScreenSize()

A function of ICNService. Call it to get your current screen resolution size. Once you get the Display Metrics information, you can adjust the position of your screen position according to the button size specified.

2. ICNImageButton Object

Every `ICNImageButton` object is corresponding to each button you see in the GUI. To use “new” to construct it, you need the current activity and **button id**. The button id is an integer number. Every button must have a unique number. When you click one button, the system service will use it to make and send bottom-down commands to your smart display

3. **void ICNElevator::addButton (ICNImageButton)**

It adds an image button to the ICNElevator manager.

4. **ICNImageButton ICNElevator::getButton(int i)**

Integer i starts from 0. This is just an index which the image button has been added to. This function gets the ICNImageButton object.

5. **void ICNImageButton::setClickedImageId(int iid)**

Integer iid is a drawable id stored in the res folder. You assign your button with a specified picture. When you click the button, it will display that picture.

6. **void ICNImageButton::setFontSize(int size)**

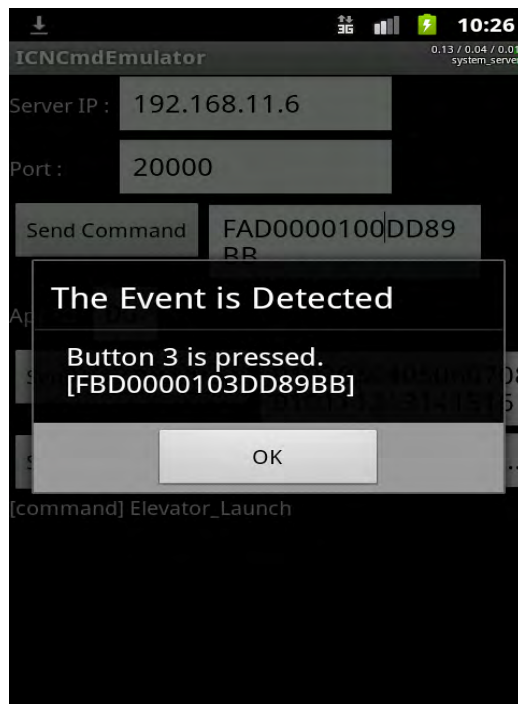
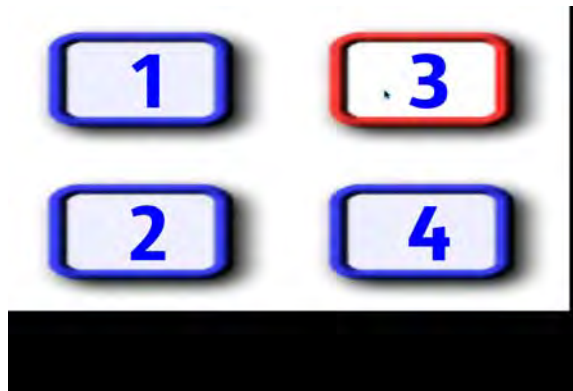
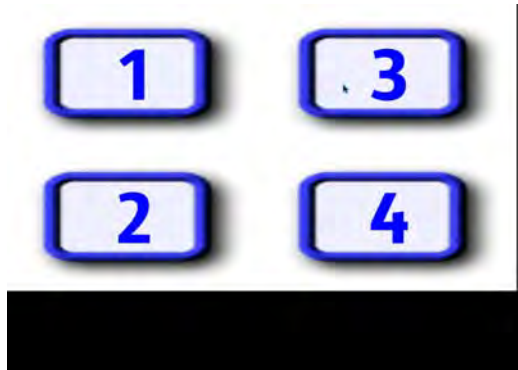
The integer size is your font size. You should use a good font size for your text or number so that it would look compatible with your button.

7. **void ICNImageButton::createButton(IBInfo ibinfo)**

Using createButton with the IBInfo object, you can generate a button body. The IBInfo object constructor needs the following input parameters:

1. *int x: the x-coordinate*
2. *int y: the y-coordinate*
3. *int w: the width of button*
4. *int h: the height of button*
5. *String img: specify the absolute file path of your background picture. (For example, /sdcard/imag/xxxx.bmp) This is an optional input. If you did not use this, you can just set it as “nul” string. Once you set it, its priority will be higher than the next parameter.*
6. *int did: it is a drawable id in your res folder of the library project. If you did not set the previous parameter, the button will use this id as its background picture.*
7. *int textcolor: This describes the color for your output.*
8. *ICNImageButton.FONT_ID fontid: you can choose the font here. We provide three fonts: ACENS, MINIS and SANSUS for you to choose from. The default font is /system/fonts/DroidSerif-Bold.ttf*
9. *String text: Your output text on ICNImageButton. You can keep this as a null string and then ICNImageButton will use the **button id** you set in the constructor time. Once you set this parameter, ICNImageButton will replace the **button id** with the text and this will appear in the output.*

- a. **If you use ICNCmdEmulator tool to launch Elevator, when you click the button “3”, you will receive a command in the ICNCmdEmulator**



- b. If you use ICNCmdEmulator tool to send a button-up command like “FAD0000203DD89BB” to the elevator, pressing the button “3” will restore

the colour to blue again.

- c. If you click the button “Send User Text (UTF-8)” to send a user text to the elevator, it will show you the text you set previously in the ICNCmdEmulator



Graphical Drawing UI

The ICNDraw Java class is a simple graphical drawing UI “activity” in the Android device. It provides you with some drawing classes such as ICNRegionView, ICNTextView, ICNImagePlayer and ICNGallery. You can rewrite your code and refer it to the onCreate() function in the SampleUserDraw.java file from your ICNMain project.

- a. The library project you need is called “ICNDraw”
- b. onCreate Function:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    rlv = new RelativeLayout(this);

    //1. Create Background View
    background = new View(this);
    background.setBackgroundResource(R.drawable.bluecurvebk);

    //2. Create Circle View 1
    circle1 = new ICNRegionView(this);
    circle1.createCircle(85, 85, 70, Color.RED, Color.TRANSPARENT,
Paint.Style.FILL_AND_STROKE);
    circle1.appearedMode(200);
    circle1.run();

    //3. Create Circle View 2
    circle2 = new ICNRegionView(this);
    circle2.setWindowRange(160, -10, 100, 100);
    circle2.createCircle(50, 50, 50, Color.WHITE, Color.TRANSPARENT,
Paint.Style.STROKE);
    circle2.appearedMode(300);
    circle2.run();

    //4. Create Rectangle View
    region1 = new ICNRegionView(this);
    Rect rect1 = new Rect(40, 40, 130, 130);
    region1.createRect(rect1, Color.YELLOW, Color.TRANSPARENT,
Paint.Style.FILL_AND_STROKE);
    region1.disappearedMode(80);
    region1.run();

    //5. Create Text View
    tv = new ICNTextView(this);
    tv.setWindowRange(50, 170, 350, 160);

    Path p = new Path();
    p.moveTo(0, 150);
```

```

p.cubicTo(0, 150, 100, 75, 350, 35);
tv.setPath(p);
tv.setText("Graphical Demo");
tv.setTextSize(40);

tv.setTextColor(Color.BLUE);
tv.setBackgroundColor(Color.TRANSPARENT);
Typeface tf = Typeface.createFromAsset(getAssets(), "fonts/bullpen3.ttf");
tv.setTypeface(tf);
tv.appearedMode(100);
tv.run();

//6. Create Round Rectangle View
region2 = new ICNRegionView(this);
Rect roundRect = new Rect(80, 400, 350, 500);
region2.createRoundRect(roundRect, Color.BLUE, Color.TRANSPARENT,
Paint.Style.FILL_AND_STROKE);
region2.appearedMode(800);
region2.run();

//6. Create Oval View
region3 = new ICNRegionView(this);
Rect roundRect2 = new Rect(80, 450, 350, 550);
region3.createOval(roundRect2, Color.GREEN, Color.TRANSPARENT,
Paint.Style.FILL_AND_STROKE);
region3.disappearedMode(500);
region3.run();

//7. Create Image Player with slide showing
imagePlayer = new ICNImagePlayer(this);

//iv.addImage(R.drawable.bluecurvebk);
imagePlayer.addImage(R.drawable.flower);
imagePlayer.addImage(R.drawable.redsky);
//imagePlayer.addImage(R.drawable.sand);
imagePlayer.addImage(R.drawable.t1);
imagePlayer.addImage(R.drawable.t2);
imagePlayer.addImage(R.drawable.sand);
imagePlayer.setWindowRange(400, 200, 600, 500);

//8. [Option] Create Gallery View for Image Player
ICNGallery mg = imagePlayer.createGallery();
mg.setWindowRange(400, 140, 600, 60);
//Start To Slide Show
imagePlayer.setSlideShowTime(3000);

try {
    imagePlayer.run();
}
catch (java.lang.OutOfMemoryError error) {
    Toast.makeText(this, "Out of Memory", 3);
}

//9. Exit button

```

```

        DisplayMetrics dm = getScreenSize();
        IBInfo ibi = new IBInfo(dm.widthPixels - 120, 0, 120, 100, "",
R.drawable.blue_blank, Color.RED, ICNImageButton.FONT_ID.MINIS, "");
        ImageButton ib = new ImageButton(this, 0);
        ib.setButtonUpAuto(true);

        ib.setFontSize(60);
        ib.createButton(ibi);
        ib.setText("Exit");

//Finally, Add All Views to Relative Layout
rlv.addView(background);
rlv.addView(circle1);
rlv.addView(circle2);
rlv.addView(region1);
rlv.addView(region2);
rlv.addView(region3);
rlv.addView(tv);
rlv.addView(imagePlayer);
rlv.addView(mg);
rlv.addView(ib);

setContentView(rlv);

}

```

1. void ICNRegionView::createCircle(float x, float y, float r, int color, int bkcolor, Paint.Style style)

To create a circle view.

1. float x: the x-coordinate of the center of the circle to be drawn
2. float y: the y-coordinate of the center of the circle to be drawn
3. int r: the radius of the circle to be drawn
4. int color: the color of the circle
5. Paint.Style style: three styles you can set.

- ◆ FILL
- ◆ STROKE
- ◆ FILL_AND_STROKE

Please refer to the android official website fore more information:

<http://developer.android.com/reference/android/graphics/Paint.Style.html>

2. void ICNRegionView::appearedMode(int ms)

When the appeared mode is set, the view area will gradually fade in from the behind. You can change the alpha values of the individual pixels .

1. int ms: Use this to control the speed of the changing alpha values. The unit is in millisecond.

3. void ICNRegionView::disappearedMode(int ms)

To set it to appeared mode, the view will gradually disappear. It works by the changing alpha value of the pixels.

2. int ms: The speed of changing alpha value can be controlled by it. The unit

is in millisecond.

4. void ICNRegionView::run()

*After calling **ICNRegionView**, it will start executing the display mode (the appeared or disappeared mode) you have set.*

void ICNRegionView::setWindowRange(int x, int y, int w, int h)

You can use this to decide the view with specified position and size in your screen. (The coordinate origin is at the upper left corner of screen)

- 1. int x: the x-coordinate of the view window*
- 2. int y: the y-coordinate of the view window*
- 3. int w: the width of the view window*
- 4. int h: the height of the view window*

Application with Drag and Drop UI

```
package icnexus.smartdisplay;

//import icnexus.sample.draganddrop.R;
import icnexus.smartdisplay.*;
import icnexus.smartdisplay.service.*;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.view.View.OnTouchListener;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.RelativeLayout.LayoutParams;

public class SampleUserDragAndDropActivity extends ICNService.Binding {
    RelativeLayout rlv;
    final int imgw = 100, imgh = 100;

    public class UserImageButton extends ICNImageButton implements
    ICNImageButtonCB {
        public UserImageButton(Activity act, int bid) {
            super(act, bid);
        }
        public void buttonDownCallback() {
            Log.v("UserImageButton", "buttonDownCallback ...");
        }
        public void buttonUpCallback() {
            Log.v("UserImageButton", "buttonUpCallback ...");
            exit();
        }
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //setContentView(R.layout.draganddrop);
        rlv = new RelativeLayout(this);
        //setContentView(rlv);

        dragPictureAndDropToTrash();
        //userDragAndDrop();
    }
}
```

```

        init(ICNService.ICN_COMMAND_ID.ICN_DRAG_AND_DROP_DEMO, rlv);

        Log.v("SampleUserDragAndDropActivity::onCreate", "bindService ...");
        bindService();
        try{
            //Waiting Service is started.
            Thread.sleep(5000);
        }
        catch(Exception e){
            e.printStackTrace();
        }
        //String className = new
String("icnexus.smartdisplay.SampleUserDragAndDropActivity");

        //sendUserData(className, userdata, 8);
        return;
    }

    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleUserDragAndDropActivity::onDestroy", "unbindService
done");

        super.onDestroy();
    }

    public void exit() {
        this.finish();
    }

    public void dragPictureAndDropToTrash() {
        ICNDragSurface dragSurface = new ICNDragSurface(this);
        dragSurface.setLayoutParams(new
LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
        ICNDragAndDropManager.getInstance().init(dragSurface);

        ImageView trashV = new ImageView(this);
        trashV.setBackgroundResource(R.drawable.empty_trash);
        LayoutParams lpv = new LayoutParams(128, 200);
        lpv.setMargins(100, 300, 0, 0);
        trashV.setLayoutParams(lpv);
        ICNDropZone dropzone1 = new ICNDropZone(trashV, dropzoneListener);

        ICNDragAndDropManager.getInstance().addDropZone(dropzone1);
    }

```

```

        ImageView i1 = new ImageView(this);
        i1.setBackgroundResource(R.drawable.android_128);
        LayoutParams lp1 = new LayoutParams(imgw, imgh);
        lp1.setMargins(300, 100, 0, 0);
        i1.setLayoutParams(lp1);

        i1.setOnTouchListener(onTouchListener);

        rlv.addView(trashV);
        rlv.addView(i1);
        rlv.addView(dragSurface);

        IBInfo ibi = new IBInfo(400, 400, 120, 100, "",
R.drawable.blue_blank, Color.RED, ICNImageButton.FONT_ID.MINIS, "");
        UserImageButton ib = new UserImageButton(this, 0);
        ib.setButtonUpAuto(true);

        ib.setFontSize(60);
        ib.createButton(ibi);
        ib.setText("Exit");

        rlv.addView(ib);

        setContentView(rlv);
    }

    private OnTouchListener onTouchListener = new OnTouchListener() {

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            if (event.getAction() == MotionEvent.ACTION_DOWN)
            {
                //String txt = ((TextView)v).getText().toString();

                //ICNDraggableItem draggedItem = new
ICNDraggableItem(v,
                //          ICNDraggableViewsFactory.getLabel("Drop
[" + txt + "]"));

                ICNDraggableItem draggedItem = new ICNDraggableItem(v,
                    ICNDraggableViewsFactory.getImage(imgw,
imgw, R.drawable.android_128));
                // Start dragging

                ICNDragAndDropManager.getInstance().startDragging(this, draggedItem);

                return true;
            }
            return false;
        }
    }

```

```

    }
};

    private ICNDDropZoneEventsListener dropzoneListener = new
ICNDDropZoneEventsListener() {
    private byte[] userdata = new byte[] {(byte)0x0f, (byte)0x0e,
(byte)0x0d, (byte)0x0c, (byte)0x0b, (byte)0x0a, (byte)0x09, (byte)0x08};

    @Override
    public void OnDragZoneEntered(ICNDDropZone zone, ICNDDraggableItem
item) {
        //zone.getView().setBackgroundColor(Color.YELLOW);
    }

    @Override
    public void OnDragZoneLeft(ICNDDropZone zone, ICNDDraggableItem
item) {
        //zone.getView().setBackgroundColor(Color.parseColor("#999999"));
    }

    @Override
    public void OnDropped(ICNDDropZone zone, ICNDDraggableItem item) {
        //zone.getView().setBackgroundColor(Color.parseColor("#999999"));
        Toast.makeText(SampleUserDragAndDropActivity.this,
"Dropped", 1000).show();
        //rlv.removeAllViews();
        rlv.removeViewAt(1);

        String className = new
String("icnexus.smartdisplay.SampleUserDragAndDropActivity");

        sendUserData(className, userdata, 8);
    }
};
}

```

API:

1. **void** OnDragZoneEntered(ICNDDropZone zone, ICNDDraggableItem item)
enter DropZone
- (1) ICNDDropZone zone : Define a zone for ICNDDrop. Any draggable item dropped to this zone will be deleted permanently.
- (2) ICNDDraggableItem item : To make an item draggable.

2. **void** OnDragZoneLeft(ICNDropZone zone, ICNDraggableItem item)
 exit DropZone

(1) ICNDropZone zone : Define a zone for ICNDrop. Any draggable item dropped to this zone will be deleted permanently.

(2) ICNDraggableItem item : To make an item draggable.

3. **void** OnDropped(ICNDropZone zone, ICNDraggableItem item)
 In the DragZone, drop the ICNDraggableItem

(1) ICNDropZone zone : Define a zone for ICNDrop. Any draggable item dropped to this zone will be deleted permanently.

(2) ICNDraggableItem item : To make an item draggable.

4. **ICNDraggableItem** ICNDraggableItem(v, ICNDraggableViewsFactory.getImage(imgw, imgh, R.drawable.android_128));

 Bind these two items: Android object onTouchListener and ImageView together so that both of them can become ICNDraggableItem at the same time.

(1) v : DraggableItem source.

(2) imgw : the width of DraggableItem

(3) imgh : the height of DraggableItem

(4) R.drawable.android_128 : To make the picture as an DraggableItem

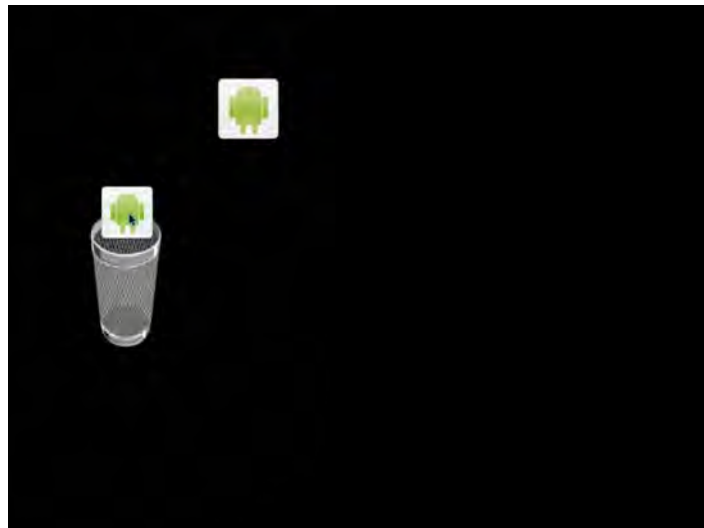
5. **static** ICNDragAndDropManager.getInstance().startDragging(this, draggedItem)
 starts dragging an draggedItem

(1) draggedItem : To make an item draggable.

6. **void** dragPictureAndDropToTrash()

 Click on a draggable item and drag it into the trash.





7. **static** ICNDragAndDropManager.getInstance().init(dragSurface)
initializes dragSurface

Digital Signage

```
package icnexus.smartdisplay;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.ArrayList;
import java.util.List;

import icnexus.smartdisplay.ICNDraw.ICNImagePlayer;
import icnexus.smartdisplay.ICNDraw.ICNTextView;
import icnexus.smartdisplay.ICNDraw.R;
import icnexus.smartdisplay.mediaplayer.VideoView;
import icnexus.smartdisplay.service.ICNService;
import android.app.Activity;
import android.graphics.Color;
import android.graphics.Path;
import android.graphics.Typeface;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.util.Log;
import android.view.ViewGroup.LayoutParams;
import android.widget.MediaController;
import android.widget.RelativeLayout;
import android.widget.Toast;

public class SampleDigitalSignage extends ICNService.Binding {
    /** Called when the activity is first created. */

    ICNTicker mTicker;
    ICNClock mClock;
    ICNImagePlayer imagePlayer;
```

```

    private TickerInfo tinfoList;
    private ClockInfo cinfoList;
    private VideoView mVideoView;
    private MediaController mMediaController;
    public RelativeLayout rlv;

    /** reading file path*/
    private String path;
    private String lpath;
    List<String> records = new ArrayList<String>();
    boolean IsRefreshPlayListRecords = false;
    boolean IsVideoStop = false;
    /** resource type*/
    private static final int LOCAL_VIDEO = 1;
    private static final int STREAM_VIDEO = 2;
    private static final int PLAYLIST_VIDEO = 3;
    private static final int PLAYLIST_VIDEO_RANDOM = 4;
    private int list_index=-1;
    private int list_num=0;
    /** video width & height*/
    private int vWidth =0;
    private int vHeight =0;

    private void TickerSetting(){

        int textColor = Color.RED, backgroundColor = Color.TRANSPARENT,
textSize = 110;
        int backgroundWidth = 800, backgroundHeight = 150;
        int x = 0, y = 0;
        int ani_begin_x = 1000, ani_end_x = 240, ani_begin_y = 580,
ani_end_y = 580;
        int ani_duration = 20000, ani_repeat_time = -1;
        String textData = "Smart Display";

        tinfoList = new TickerInfo(textColor, backgroundColor, textSize,
backgroundWidth, backgroundHeight,
x, y,
ani_begin_x, ani_end_x, ani_begin_y, ani_end_y,
ani_duration, ani_repeat_time,
textData);
    }

    private void TickerDataSetting(String text){
        mTicker.TextSetting(text);
    }

    private void ClockSetting(){

```

```

        int window_x = 0, window_y = 615, window_width = 240,
window_height = 75;
        int size = 50, textColor = Color.GREEN, backgroundColor =
Color.DKGRAY;
        Typeface tf = Typeface.MONOSPACE;

        cinfoList = new ClockInfo(window_x, window_y, window_width,
window_height,
            size, textColor, backgroundColor, tf);
    }

    private void ImagePlayerSetting(){

        imagePlayer.addImage(R.drawable.p1);
        imagePlayer.addImage(R.drawable.p2);
        imagePlayer.addImage(R.drawable.p3);
        imagePlayer.addImage(R.drawable.p4);
        imagePlayer.addImage(R.drawable.p5);
        imagePlayer.setWindowRange(350, 60, 600, 500);
        imagePlayer.setSlideShowTime(6000);

        try {
            imagePlayer.run();
        }
        catch (java.lang.OutOfMemoryError error) {
            Toast.makeText(this, "Out of Memory", 3);
        }
    }

    /** video play */
    private void playback(){
        if (path == "") {
            // Tell the user to provide a media file URL/path.
            Toast.makeText(
                this,
                "Please edit VideoViewDemo Activity, and set path"
                + " variable to your media file URL/path",
                Toast.LENGTH_LONG).show();
        } else {

            /*
             * Alternatively, for streaming media you can use
             * mVideoView.setVideoURI(Uri.parse(URLstring));
             */
            resetMedia(3);
            //mVideoView.setVideoPath(path);
            mVideoView.setMediaController(mMediaController);
        }
    }

```

```

        mVideoView.requestFocus();
    }
}
/** video path issue */
private void resetMedia(Integer Media){

    /**
     * @Parameter: Media - play data from which type
     */

    setMediaVideoPath(Media);

    try {
        mVideoView.setVideoPath(path);
    } catch (IllegalStateException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

private void setMediaVideoPath(Integer Media) {
    try {

        switch (Media) {
            case LOCAL_VIDEO:
                /*
                 * TODO: Set the path variable to a local media file path.
                 */
                path = "sdcard/video/BBCexample.flv";
                if (path == "") {
                    // Tell the user to provide a media file URL.
                    Toast.makeText(
                        this,
                        "Please edit MediaPlayerDemo_Video
Activity, "
                        + "and set the path variable
to your media file path."
                        + " Your media file must be
stored on sdcard.",
                        Toast.LENGTH_LONG).show();
                }
                break;
            case STREAM_VIDEO:
                path = "";
                if (path == "") {
                    // Tell the user to provide a media file URL.
                    Toast
                        .makeText(
                            this,

```

```

Activity,"
                                "Please edit MediaPlayerDemo_Video
                                + " and set the path variable
to your media file URL.",
                                Toast.LENGTH_LONG).show();
    }

    break;
case PLAYLIST_VIDEO:
    /**
     * Refresh Playlist records Once (maybe limit N lines)
     *
     */
    if (IsRefreshPlayListRecords){
        break;
    }else{
        IsRefreshPlayListRecords = false;
    }

    lpath = "playlist.lst";
    File f = new File("/mnt/sdcard/playlist", lpath);
    int c = -1 ;
    int new_choice = 0;
    if(f.exists()){
        //StringBuffer contents = new StringBuffer();
        BufferedReader buffreader = null;
        try {
            // open file for reading
            buffreader = new BufferedReader(new
FileReader(f));

            String line = null;
            /**
             * buffreader.mark(1000); useless
             */

            // init.
            list_index = -1;

            if (!records.isEmpty()){
                records.clear();
                Log.d("SampleDigitalSignage::MediaPlayer",
"debug: release the playlist records");
            }

            // read every line of the file into the line-
variable, on line at the time
            while (( line = buffreader.readLine()) != null)
            {
                /**if (c > list_index){

```

```

// mark.shieh 2011.11.7 record read
line_num. and catch the path

        list_index = c;
        path = line;
        new_choice =1;
        break;
    }*/
    records.add(line);
    c ++;
}

if (c > list_index){
    list_index = c;
}

} catch (FileNotFoundException e) {
    Toast.makeText(
        this,
        "Playlist Gone, please check file.",
        Toast.LENGTH_SHORT).show();
    e.printStackTrace();
} catch (java.io.IOException e) {
    Toast.makeText(
        this,
        "Some thing wrong.",
        Toast.LENGTH_SHORT).show();
    e.printStackTrace();
} finally {
    try {
        if (buffreader != null){
            buffreader.close();
        }
    } catch (java.io.IOException e) {
        Toast.makeText(
            this,
            "Buffer for Reading can not release.",
            Toast.LENGTH_SHORT).show();
        e.printStackTrace();
    }
}
} else{
    Toast.makeText(
        this,
        "Please add the playlist first.",
        Toast.LENGTH_SHORT).show();
}
break;

```

```

    }

    /**
     * Get Path from list
     *
     * Important!! List.get(N) must change DataType, because return
value is Object
     */
    try{
        if(list_num<=list_index){
            path = (String)records.get(list_num);
            list_num ++;
        }else{
            path = (String)records.get(0);
            list_num =1;
        }
    } catch (IndexOutOfBoundsException e){
        Log.e("SampleDigitalSignage::MediaPlayer", "select " +
list_num + "/" + list_index + " error: " + e.getMessage(), e);
    }

    /** mark.shieh 2011.11.3*/
    //mediaController = new MediaController(this);
    } catch (Exception e) {
        Log.e("SampleDigitalSignage::MediaPlayer", "error: " +
e.getMessage(), e);
    }
}
/** video info. record*/
private void init_video_info_record(){
    vWidth =mVideoView.getWidth();
    vHeight =mVideoView.getHeight();
}

private void startplay(){
    mVideoView.start();
}

private void pauseplay(){
    mVideoView.pause();
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    TickerSetting();
    ClockSetting();

```

```

        rlv = new RelativeLayout(this);
        imagePlayer = new ICNImagePlayer(this);
        ImagePlayerSetting();
        mTicker = new ICNTicker(this, tinfoList);

        mClock = new ICNClock(this, cinfoList);
        mVideoView = new VideoView(this);

        //////////////////////////////////////

        mMediaController = new MediaController(this);
        mMediaController.setAnchorView(mVideoView);

        mVideoView.setOnCompletionListener(new OnCompletionListener(){
            @Override
            public void onCompletion(MediaPlayer arg0) {
                Log.d("SampleMediaPlayer", "video [" + path + "] finish");
                resetMedia(3);
            }
        });

        mVideoView.setOnPreparedListener(new MediaPlayer.OnPreparedListener()
        {

            @Override
            public void onPrepared(MediaPlayer mp) {
                // TODO Auto-generated method stub
                Log.d("SampleMediaPlayer", "onPrepared");
                init_video_info_record();
                mVideoView.start();
            }
        });

        mVideoView.ChangeDisplayMode(0);

        /** prepare the play op.*/
        playback();
        //////////////////////////////////////

        rlv.addView(mClock);
        rlv.addView(mVideoView);
        rlv.addView(mTicker);
        rlv.addView(imagePlayer);
        setContentView(rlv);

        init(ICNService.ICN_COMMAND_ID.ICN_DIGITAL_SIGNAGE_DEMO, rlv);
        Log.v("SampleDigitalSignage:onCreate", "bindService ...");

```

```

        bindService();
    }

    public void stringArrivalCallback(char[] userdata, int datasize) {
        String str = new String(userdata);
        //Log.v("SampleUserElevator::stringArrivalCallback::User String=",
str + " " + str.length());
        //String str_tmp = "天啊";
        TickerDataSetting(str);
    }

    public void playCallback(){
        if(IsVideoStop == false)
            mVideoView.start();
        else
            playback();

        IsVideoStop = false;
    }

    public void pauseCallback(){
        mVideoView.pause();
    }

    public void stopCallback(){
        mVideoView.stopPlayback();
        IsVideoStop = true;
    }

    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleDigitalSignage::onDestroy", "unbindService done");

        super.onDestroy();
    }
}

```

API:

1. **void** TickerInfo (textColor, backgroundColor, textSize, backgroundWidth, backgroundHeight, x, y, ani_begin_x, ani_end_x, ani_begin_y, ani_end_y, ani_duration, ani_repeat_time, textData)

The parameter definition for Ticker's values

- (1) **int** textCoLoR : Ticker's text color
- (2) **int** backgroundColor : Ticket's background color
- (3) **int** textSize : Ticker's font size
- (4) **int** backgroundWidth : the width of Ticker's background
- (5) **int** backgroundHeight : the height of Ticker's background
- (6) **int** x : Ticker's x coordinate of the upper-left corner
- (7) **int** y : Ticker's y coordinate of the upper-left corner
- (8) **int** ani_begin_x : Ticker's x coordinate of the starting point at the upper-left corner
- (9) **int** ani_end_x : Ticker's x coordinate of the end point at the upper-left corner
- (10) **int** ani_begin_y : Ticker's y coordinate of the starting point at the upper-left corner
- (11) **int** ani_end_y : Ticker's y coordinate of the end point at the upper-left corner
- (12) **int** ani_duration : the time Ticker takes to complete the whole path (ms)
- (13) **int** ani_repeat_time : the number of times Ticker has run. -1 implies infinity
- (14) **String** textData : the text content of Ticker

2. **void** ICNTicker.TextSetting (String text)

Define Ticker's text

3. **void** ClockInfo (window_x, window_y, window_width, window_height, size, textCoLoR, backgroundColor, tf)

The parameter definition for Clock's values

- (1) **int** window_x : Clock's x coordinate of the upper left corner of the background
- (2) **int** window_y : Clock's y coordinate of the upper left corner of the background
- (3) **int** window_width : the width of Clock's background
- (4) **int** window_height : the height of Clock's background
- (5) **int** size : Clock's font size
- (6) **int** textCoLoR : Clock's font color
- (7) **int** backgroundColor : Clock's background color
- (8) Typeface tf : Clock's text font type

4. **void** ICNImagePlayer.addImage (image_id)

Add the image to the image player

5. **void** setWindowRange(int x, int y, int w, int h)

Specify the location, length and width for the image player

- (1) x: The image player's x coordinate of the upper left corner.
- (2) y: The image player's y coordinate of the upper left corner.
- (3) w: the width of the image player
- (4) h : the height of the image player

6. **void** setSlideShowTime(int time)

- (1) time: the time the image player takes to replace a picture(ms)

7. ICNImagePlayer.run()

Set when the image player starts operating

Chart

```
package icnexus.smartdisplay;

import org.achartengine.GraphicalView;

import android.os.Handler;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.widget.RelativeLayout;
import android.widget.RelativeLayout.LayoutParams;

import icnexus.smartdisplay.IDemoChart;
import icnexus.smartdisplay.SalesGrowthChart;
import icnexus.smartdisplay.ChartInfo;
import icnexus.smartdisplay.SampleUserDragAndDropActivity.UserImageButton;
import icnexus.smartdisplay.service.*;

import java.util.Vector;

public class SampleChart extends ICNService.Binding {
    //private IDemoChart[] mCharts = new IDemoChart[] {new SalesGrowthChart()};
    private IDemoChart mCharts = new SalesGrowthChart();

    //private String[] mMenuText;

    //private String[] mMenuSummary;
    private Handler handler = new Handler();

    private ChartInfo cinfoList;

    private Vector<Double> XAxe = new Vector<Double>();
    private Vector<Double> YAxe = new Vector<Double>();
    private Vector<Double> DataYAxe = new Vector<Double>();

    String ChartTitle = "Chart Demo", XName = "X", YName = "Y";
    String ChartExplain = "Random value chart diagram";
    double XMin = 0, XMax = 20, YMin = 0, YMax = 15;
    int ColorAxes = Color.LTGRAY, ColorLabel = Color.YELLOW;
    int ColorChart = Color.RED, ColorBackgground = Color.rgb(100, 50, 200);
    int XRangeNum = 5, YRangeNum = 5;
    boolean ShowGrid = false, BackgroundColorApply = true;
    boolean ZoomButtonsVisible = false, ShowLegend = false;
    boolean ShowAxes = true, ShowLabels = true;
```

```

float ChartSmoothness = 0.33f;

int XAxisNumber = 21;
int YAxisNumber = 21;
int ChartRefreshTime = 500;

public GraphicalView cChartView;
public RelativeLayout rlv;
protected LayoutParams mlp;

int InputDataSize=0;

IBInfo ibi;
ImageButton ib;

public void info_update()
{
    cinfoList = new ChartInfo(XMin, XMax, YMin, YMax, ColorAxes,
ColorLabel,
                                ColorChart, ColorBackgground, XRangeNum, YRangeNum,
ChartSmoothness,
                                ShowGrid, BackgroundColorApply, ZoomButtonsVisible,
ShowLegend,
                                ShowAxes, ShowLabels,
                                ChartTitle, XName, YName, ChartExplain);
}

public Runnable showTime = new Runnable() {
    public void run()
    {
        chart_update();
        handler.postDelayed(this, ChartRefreshTime);
    }
};

public void ReadInputData(byte[] userdata, int datasize)
{
    InputDataSize = datasize;
    for(int i = 0; i < datasize; i++)

DataYAxis.addElement(Double.valueOf(Integer.toString(userdata[i])));
}

public void chart_update()
{
    //YAxis.clear();
    YAxis.remove(0);

    if(InputDataSize > 0)

```

```

{
    YAxie.addElement(DataYAxie.elementAt(0));
    InputDataSize--;
    DataYAxie.remove(0);
}
else
{
    //for(int j=0; j<YAxieNumber; j++)
    //{
    //    YAxie.addElement(Math.random() * 10);
    //}
    //YAxie.addElement(Math.random() * 10);
    YAxie.addElement((double)5);
}

//cChartView = mCharts[0].change(this, cinfoList, XAxie, YAxie);
cChartView = mCharts.change(this, cinfoList, XAxie, YAxie);
mlp = new LayoutParams(500, 350);
mlp.setMargins(250, 150, 0, 0);
cChartView.setLayoutParams(mlp);
rlv = new RelativeLayout(this);
//////////exit button//////////
ibi = new IBInfo(800, 400, 120, 100, "", R.drawable.blue_blank,
Color.RED, ICNImageButton.FONT_ID.MINIS, "");
ib = new UserImageButton(this, 0);
ib.setButtonUpAuto(true);

ib.setFontSize(60);
ib.createButton(ibi);
ib.setText("Exit");
//////////
rlv.addView(cChartView);
rlv.addView(ib);
setContentView(rlv);
}

public class UserImageButton extends ICNImageButton implements
ICNImageButtonCB {
    public UserImageButton(Activity act, int bid) {
        super(act, bid);
    }
    public void buttonDownCallback() {
        Log.v("UserImageButton", "buttonDownCallback ...");
    }
    public void buttonUpCallback() {
        Log.v("UserImageButton", "buttonUpCallback ...");
        exit();
    }
}

```

```

public void exit() {
    this.finish();
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    rlv = new RelativeLayout(this);

    //int length = mCharts.length;
    //mMenuText = new String[length + 2];
    //mMenuSummary = new String[length + 2];
    //mMenuText[0] = "Embedded chart demo";
    //mMenuSummary[0] = "A demo on how to include a chart into a graphical
activity";
    //for (int i = 0; i < length; i++) {
    //    mMenuText[i + 1] = mCharts[i].getName();
    //    mMenuSummary[i + 1] = mCharts[i].getDesc();
    //}
    //mMenuText[length + 1] = "Random values charts";
    //mMenuSummary[length + 1] = "Chart demos using randomly generated
values";

    info_update();

    for(int i=0; i<XAxisNumber; i++)
    {
        XAxis.addElement((double)i);
    }
    for(int j=0; j<YAxisNumber; j++)
    {
        YAxis.addElement(Math.random() * 10);
    }

    //cChartView = mCharts[0].change(this, cinfoList, XAxis, YAxis);
    cChartView = mCharts.change(this, cinfoList, XAxis, YAxis);
    mlp = new LayoutParams(500, 350);
    mlp.setMargins(250, 150, 0, 0);
    cChartView.setLayoutParams(mlp);

    //exit button
    ibi = new ImageButton(800, 400, 120, 100, "", R.drawable.blue_blank,
Color.RED, ICNImageButton.FONT_ID.MINIS, "");
    ib = new ImageButton(this, 0);
    ib.setButtonUpAuto(true);

```

```

        ib.setFontSize(60);
        ib.createButton(ibi);
        ib.setText("Exit");
        //////////////////////////////////////

        rlv.addView(cChartView);
        rlv.addView(ib);

        setContentView(rlv);

        handler.postDelayed(showTime, ChartRefreshTime);

        init(ICNService.ICN_COMMAND_ID.ICN_CHART_DEMO, rlv);
        Log.v("SampleChart::onCreate", "bindService ...");
        bindService();

    }

    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        unbindService();
        Log.v("SampleChart::onDestroy", "unbindService done");

        super.onDestroy();
    }

    public void dataArrivalCallback(byte[] userdata, int datasize) {
        Log.v("SampleChart::dataArrivalCallback",
            Integer.toString(datasize));

        //for(int i = 0; i < datasize; i++)
        //Log.v("SampleChart::dataArrivalCallback::User Data=",
            String.format("%#x", userdata[i]));

        ReadInputData(userdata, datasize);
    }
}

```

API:

1. ChartInfo.ChartInfo(double cXMin, double cXMax, double cYMin, double cYMax, int cColorAxes, int cColorLabel, int cColorChart, int cColorBackgground, int cXRangeNum, int cYRangeNum, float cChartSmoothness, boolean cShowGrid, boolean cBackgroundColorApply, boolean cZoomButtonsVisible, boolean cShowLegend, boolean cShowAxes, boolean cShowLabels, String cChartTitle, String cXName, String cYName, String cChartExplain)

Parameter definition for Chart:

- (1) **XMin** : The minimum Chart's x-axis value
- (2) **XMax** : The maximum Chart's x-axis value
- (3) **YMin** : The minimum Chart's y-axis value
- (4) **YMax** : The maximum Chart's y-axis value
- (5) **ColorAxes** : Chart's axes color
- (6) **ColorLabel** : Chart's label color
- (7) **ColorChart** : Chart's line color
- (8) **ColorBackgground** : Chart's background color
- (9) **XRangeNum** : The number of Chart's x-axis intervals
- (10) **YRangeNum** : The number of Chart's y-axis intervals
- (11) **ChartSmoothness** : Chart's line smoothness
- (12) **ShowGrid** : The Boolean value to specify whether to display or hide gridlines in Chart
- (13) **BackgroundColorApply** : The Boolean value to specify whether the background color is applied to Chart. A false means 'transparent' background and a true will change the ColorBackgground to the background color
- (14) **ZoomButtonsVisible** : The Boolean value to specify whether the zoom interface is displayed
- (15) **ShowLegend** : The Boolean value to specify whether Chart's text description at the bottom left corner is displayed
- (16) **ShowAxes** : The Boolean value to specify whether Chart's axes are displayed.
- (17) **ShowLabels** : The Boolean value to specify whether the label names of Chart's axes are displayed.
- (18) **ChartTitle** : The Boolean value to specify whether Chart's title name is displayed
- (19) **XName** : The name of Chart's x-axis
- (20) **YName** : The name of Chart's y-axis
- (21) **ChartExplain** : Chart's text description at the bottom left corner

```
2. IDemoChart.change(Context context, ChartInfo ci, Vector<Double> XAxie,  
Vector<Double> YAxie)
```

Add parameter Context, ChartInfo and the information of x axis and the y axis.

MediaPlayer API

1. Introduction

This section illustrates how the SampleMediaPlayer works, with a simple guide to demonstrate the development of MediaPlayer.

2. Building an architecture for the system

This section will divide into two parts. In the first part, you will be introduced to a number of ICNMediaPlayer's playback related-settings that adjusts the video size and display resolution. In the second part, you will learn to start SampleMediaPlayer on specified time and specify the position of the media player.

3. Basic function

Section 3 gives the description of SampleMediaPlayer.

3.1 Execution

```
package    icnexus.smartdiaplay;
import     icnexus.smartdisplay.mediaplayer.VideoView;
import     android.app.Activity;
import     android.os.Bundle;
import     android.util.Log;
```

To control and use a certain media data type:

```
import     android.media.MediaPlayer;
import     android.media.MediaPlayer.OnCompletionListener;
import     android.widget.MediaController;
import     android.widget.MediaController.MediaPlayerControl;
```

Use the code below if you would like to connect to the Serial Port or Com Port to enable remote control:

```
import     icnexus.smartdisplay.service.ICNService;
import     android.app.Activity;
```

First of all, you need to create a class that extends an activity. Then, you implement the MediaPlayerControl. If you have remote control feature, you could create a SampleMediaPlayer class that extend ICNService.Binding instead:

```
public class SampleMediaPlayer extends ICNService.Binding implements
MediaPlayerControl{
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

Then, you can set up the required objects.

```
private static final String TAG = "DEMO"
private VideoView mVideoView;
private MediaController mMediaController;
```

In the onCreate() function, add the feature and listener status to the VideoView object and MediaController object. For example,

```
VideoView :
VideoView.setOnCompletionListener ( new OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer arg0){ ... } } );
VideoView.setOnPreparedListener ( new OnPreparedListener() {
    @Override
    public void onPrepared(MediaPlayer arg0){ ... } } );
VideoView.setOnErrorListener ( new OnErrorListener() {
    @Override
    public void onError(MediaPlayer arg0, int what, int extra){ ... }
} );
```

ErrorListener is an optional item. You do not have to customize or respond to an error message.

```
MediaController :
MediaController.setPrevNextListeners(ClickListener Next, ClickListener Prev);
```

Users must customize their own button method of cache replacement for streaming media. Below is an example:

```
MediaController.setPrevNextListeners(new View.OnClickListener() {
    @Override
    public void onClick(View v) { ... }
}, new View.OnClickListener() {
    @Override
    public void onClick(View v) {... } } );
```

It is recommended to call the onDestroy() method to stop the process so that the service is effectively terminated.

```
@Override
protected void onDestroy() { ... }
```

Above is the basic information you need to know to start the development of SimpleMediaPlayer. For further information, please visit the source code.

3.2 Video size and position

To specify the position of the video, use RelativeLayout. Call the method inside the VideoView class to modify the video size.

```
import android.widget.RelativeLayout;
```

Declare a layout container and its stored location in the VideoView class.

```
private RelativeLayout rlv;  
RelativeLayout.LayoutParams rlt = null;
```

Create an object. Set the image size as WRAP_CONTENT for now. You may change its name later.

```
rlv = new RelativeLayout(this);  
rlt = new RelativeLayout.LayoutParams(  
RelativeLayout.LayoutParams.WRAP_CONTENT,  
RelativeLayout.LayoutParams.WRAP_CONTENT);
```

Set the position of the media player. setMargins(int left, int top, int bottom, int right) is to set the margins in pixels.

```
rlt.setMargins(50, 200, 0, 0);
```

Remember to store the rlt values in the VideoView object.

```
mVideoView.setLayoutParams(rlt);
```

There are four video display modes: the origin/natural size, full screen (fixed length to width ratio), custom size screen (fixed length to width ratio) and free size (you may set the length and the width any form you like). The video display modes range from 0 to 3. It is recommended that you choose 2 (custom size screen).

```
mVideoView.ChangeDisplayMode(2);  
mVideoView.setVideoScale(720, 405);
```

Now you can add the VideoView to your Layout. Then declare it as Class.

```
rlv.addView(mVideoView);  
setContentView(rlv);
```

The above explains how you can specify the position of and resize your video. Part of the methods are created and owned by ICNexus. If you need other functions, you may also try to edit on your own.

3.3 檔 Playback control

This section tells you how to control playback of your media files, such as setting a looping order, play/pause and next/previous.

Here we will use some examples which you can apply to a variety of tasks, such as modifying your playback sequence, adding more media files during the running time and exchanging media files.

Below are the examples. Please note that we did not provide the full source code. If you would like to view the full code, please refer to your JAR files.

```
a. private void playback(void)  
b. private void resetMedia(Integer Media)
```

c. `private void setMediaVideoPath(Integer Media)`

- a. Confirm the path for the file. This process appears once every time you start the program. You may also call it “initialization check”.
- b. This will be called after the video starts playing. It will also be called by `OnCompleteListener` when the video finishes playing.
- c. The sub-function of `resetMedia`. It is a two-step process for updating video addresses. First, you must save a specific text file path into a list. The list will provide you with the ability to modify your playback in a dynamic way. In the example above, we have mentioned about adding playback component. This process will only repeat once unless you re-read the file’s playlist. Second, once the playlist has been saved as a List, this method will update the playback file location.

We will not go into the detailed discussion of `VideoView` and `MediaController`. Here you will learn basic principles of the playback controls, such as play, pause, changing timelines and stop.

Name	Description
<code>public void start()</code>	Starts or resumes playback.
<code>public void pause()</code>	Pauses playback.
<code>public int getDuration()</code>	Gets the duration of the current media file.
<code>public void seekTo(int pos)</code>	Seeks to specified time position. The offset is in milliseconds from the start to seek to. Must be smaller than the total duration.
<code>public int getCurrentPosition()</code>	Gets the current playback position.
<code>public int getBufferPercentage()</code>	The percentage of the current Video that has been buffered.

To exit the player completely, you must close the player window or call the `stopCallback` method via remote control. If you would like to stop playing the current media file only, please use `Pause` or other restart methods.

3.4 Communicating with service

Please add the following setting to the `onCreate()` method to complete the declaration of remote control and Layout container.

```
init(ICNService.ICN_COMMAND_ID.ICN_Media_Player_DEMO, rlv);  
bindService();
```

The service implementation examples include seven methods, control functions and corresponding functions as shown below.

<code>playCallback()</code>	<code>start()</code>
<code>stopCallback()</code>	<code>onDestroy()</code>
<code>pauseCallback()</code>	<code>pause()</code>

nextCallback()	setPreNextListeners
prevCallback()	setPreNextListeners
fastForwardCallback()	copy from MediaController
fastBackCallback()	copy from MediaController

If you would like to add new commands, please see the Service section.

You may start the player to send and receive information through COM or Network. It requires the following commands.

ICN Demo Command	Start Byte	Command Bytes	Data Byte	End Bytes
Media Player 1. Launch Video	0xFA	0xD0 0x05 0x01	0x00	0xDD 0x89 0xBB
2. Play/Resume Video	0xFA	0xD0 0x05 0x02	0x00	0xDD 0x89 0xBB
3. Pause Video	0xFA	0xD0 0x05 0x03	0x00	0xDD 0x89 0xBB
4. Stop Video	0xFA	0xD0 0x05 0x04	0x00	0xDD 0x89 0xBB
5. Video Forward	0xFA	0xD0 0x05 0x05	0x00	0xDD 0x89 0xBB
6. Video Backward	0xFA	0xD0 0x05 0x06	0x00	0xDD 0x89 0xBB
7. Next Video	0xFA	0xD0 0x05 0x07	0x00	0xDD 0x89 0xBB
8. Prev Video	0xFA	0xD0 0x05 0x08	0x00	0xDD 0x89 0xBB

We also provide remote control playback capabilities. All you need is put the file in the path after: /mnt/sdcard/video or /mnt/sdcard/DCIM, give a filename and your device will automatically save the file as readable format. The command is:

Send User Data Bytes To Smart Display Data Stream (16 bytes)	0xFA	0x05 0xFF 0x00	0x10 [16 bytes]	0xDD 0x89 0xBB
---	------	----------------	-----------------	----------------

Please convert your file name into hexadecimal encoding before sending the commands to the machine. You may follow the procedure below:

(Conversion tool: You can send commands from ICNCmdEmulator to your NSD or Android Virtual Device via the network. The character will automatically be encoded. You must encode your own Com port. You could use Notepad++ or Ultraedit for converting before sending the hexadecimal information via Docklight.

Eg :

Filename :	Sample.mp4
After conversion :	53 61 6D 70 6C 65 2E 6D 70 34
Length :	00 0A
MediaPlayer ID number :	05
Start :	FA
Finsh :	DD 89 BB
Last data tranmssion :	FA 05 FF 00 0A 53 61 6D 70 6C 65 2E 6D 70 34 DD 89 BB

ResourceUsage API

The ResourceUsage Java class is a simple graphical resource usage of CPU and memory UI “activity” in the Android device. It shows you how to complete a simple resource usage UI with some drawing classes such as as ICNRegionView, ICNTextView. You can rewrite your code in the ICNUsageGUI.java file from your ICNMain project.

```
package icnexus.smartdisplay;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.ArrayList;

import icnexus.smartdisplay.ICNDraw.ICNRegionView;
import icnexus.smartdisplay.ICNDraw.ICNTextView;
import icnexus.smartdisplay.service.ICNService;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.os.Bundle;
import android.os.Handler;
import android.util.DisplayMetrics;
import android.widget.RelativeLayout;

public class ICNUsageGUI extends ICNService.Binding {
    /** Called when the activity is first created. */
    ArrayList<ICNRegionView> cpuUsageGUIList = new ArrayList<ICNRegionView>();
    ICNRegionView cpuUsageBk;
    ICNTextView cpuTitleText, cpuUsageText;
```

```

        ArrayList<ICNRegionView> memoryUsageGUIList = new
ArrayList<ICNRegionView>();
        ICNRegionView memoryUsageBk;
        ICNTextView memoryTitleText, memoryUsageText, memorySizeText;

        final int LIGHT_LEVEL = 20;
        int LIGHT_WIDTH = 50;
        int LIGHT_HEIGHT = 10;
        int LIGHT_DISTANCE_Y;
        int LIGHT_DISTANCE_X;
        int USAGE_TEXT_SCOPE_WIDTH;
        int USAGE_TEXT_SCOPE_HEIGHT;
        int USAGE_TITLE_SCOPE_HEIGHT;
        int USAGE_SCOPE_EDGE = 15;

        long Mem_Used = 0;

        int screen_width, screen_height;

        public RelativeLayout rlv_main;

        private DisplayMetrics dm;

        private int RefreshTime = 1500;

        private Handler handler = new Handler();

        public static enum ICN_MEM_UNIT {
            ICN_MEM_KB(-1),
            ICN_MEM_MB(0),
            ICN_MEM_GB(1),
            ICN_MEM_TB(2);

            private int value;

            private ICN_MEM_UNIT(int value) {
                this.value = value;
            }

            static ICN_MEM_UNIT fromValue(int value) {
                for (ICN_MEM_UNIT my: ICN_MEM_UNIT.values()) {
                    if (my.value == value) {
                        return my;
                    }
                }

                return null;
            }

            int value() {
                return value;
            }

            public String toString() {
                switch (fromValue(value)) {
                    case ICN_MEM_KB:
                        return new String("KB");

```

```

        case ICN_MEM_MB:
            return new String("MB");
        case ICN_MEM_GB:
            return new String("GB");
        case ICN_MEM_TB:
            return new String("TB");
        default :
            return new String("");
    }
}
}
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    rlv_main = new RelativeLayout(this);
    //get screen size
    dm = new DisplayMetrics();
    getWindowManager().getDefaultDisplay().getMetrics(dm);
    screen_width = dm.widthPixels;
    screen_height = dm.heightPixels;

    LIGHT_WIDTH = screen_width / 6;
    LIGHT_HEIGHT = screen_height / 50;
    LIGHT_DISTANCE_Y = (LIGHT_HEIGHT * 3 / 2);
    LIGHT_DISTANCE_X = (LIGHT_WIDTH + 5);
    USAGE_TEXT_SCOPE_WIDTH = LIGHT_WIDTH * 2 + 5;
    USAGE_TEXT_SCOPE_HEIGHT = screen_height / 7;
    USAGE_TITLE_SCOPE_HEIGHT = screen_height * 60 / 480;

    //Log.v("SampleChart", "Screen Info:" + dm.widthPixels + " / " +
dm.heightPixels);

    createCPUUsageGUI(rlv_main, screen_width / 9, 10);
    createMemoryUsageGUI(rlv_main, screen_width * 5 / 9, 10);
    updateCPUUsage((int)readCPUUsage());
    updateMemoryUsage((int)readMEMUsage());
    updateUsedMemorySize(Mem_Used, ICN_MEM_UNIT.ICN_MEM_KB);
    setContentView(rlv_main);

    handler.postDelayed(info_update_r, RefreshTime);

    init(ICNService.ICN_COMMAND_ID.ICN_RESOURCE_USAGE, rlv_main);
    bindService();
}

@Override
protected void onDestroy() {
    // TODO Auto-generated method stub

    unbindService();
    super.onDestroy();
}

public Runnable info_update_r = new Runnable() {
    public void run()
    {

```

```

        info_update();
        handler.postDelayed(this, RefreshTime);
    }
};

public void info_update()
{
    updateMemoryUsage((int)readMEMUsage());
    updateCPUUsage((int)readCPUUsage());
    updateUsedMemorySize(Mem_Used, ICN_MEM_UNIT.ICN_MEM_KB);
}

public void createCPUUsageGUI(RelativeLayout rlv, int x, int y) {

    cpuUsageBk = new ICNRegionView(this);
    Rect rectBk = new Rect(x, y, x + LIGHT_WIDTH + LIGHT_DISTANCE_X +
USAGE_SCOPE_EDGE * 2, USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT +
LIGHT_LEVEL * LIGHT_DISTANCE_Y + USAGE_TEXT_SCOPE_HEIGHT);
    cpuUsageBk.createRect(rectBk, Color.BLACK, Color.TRANSPARENT,
Paint.Style.FILL_AND_STROKE);
    rlv.addView(cpuUsageBk);

    cpuTitleText = new ICNTextView(this);
    cpuTitleText.setWindowRange(x, y, LIGHT_WIDTH + LIGHT_DISTANCE_X +
USAGE_SCOPE_EDGE * 2, USAGE_TITLE_SCOPE_HEIGHT);
    cpuTitleText.setText("CPU Usage");
    cpuTitleText.setTextSize(screen_height * 26 / 480);
    cpuTitleText.setTextColor(Color.YELLOW);
    cpuTitleText.setBackgroundColor(Color.TRANSPARENT);
    rlv.addView(cpuTitleText);

    for (int i = 0; i < LIGHT_LEVEL; i++) {
        cpuUsageGUIList.add(new ICNRegionView(this));
        Rect rectLeft = new Rect(USAGE_SCOPE_EDGE + x,
USAGE_TITLE_SCOPE_HEIGHT + y + i * LIGHT_DISTANCE_Y, USAGE_SCOPE_EDGE + x +
LIGHT_WIDTH, USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + i *
LIGHT_DISTANCE_Y);
        cpuUsageGUIList.get(2 * i).createRect(rectLeft, Color.GREEN,
Color.TRANSPARENT, Paint.Style.FILL_AND_STROKE);
        rlv.addView(cpuUsageGUIList.get(2 * i));

        cpuUsageGUIList.add(new ICNRegionView(this));
        Rect rectRight = new Rect(USAGE_SCOPE_EDGE + x +
LIGHT_DISTANCE_X, USAGE_TITLE_SCOPE_HEIGHT + y + i * LIGHT_DISTANCE_Y,
USAGE_SCOPE_EDGE + x + LIGHT_WIDTH + LIGHT_DISTANCE_X,
USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + i * LIGHT_DISTANCE_Y);
        cpuUsageGUIList.get(2 * i + 1).createRect(rectRight,
Color.GREEN, Color.TRANSPARENT, Paint.Style.FILL_AND_STROKE);
        rlv.addView(cpuUsageGUIList.get(2 * i + 1));
    }

    cpuUsageText = new ICNTextView(this);
    cpuUsageText.setWindowRange(USAGE_SCOPE_EDGE + x,
USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + LIGHT_LEVEL * LIGHT_DISTANCE_Y,
USAGE_TEXT_SCOPE_WIDTH, USAGE_TEXT_SCOPE_HEIGHT);
    cpuUsageText.setTextSize(screen_height * 30 / 480);
    cpuUsageText.setTextColor(Color.GREEN);

```

```

        cpuUsageText.setBackgroundColor(Color.TRANSPARENT);

        rlv.addView(cpuUsageText);
    }

    public void createMemoryUsageGUI(RelativeLayout rlv, int x, int y) {

        memoryUsageBk = new ICNRegionView(this);
        Rect rectBk = new Rect(x, y, x + LIGHT_WIDTH + LIGHT_DISTANCE_X +
        USAGE_SCOPE_EDGE * 2, USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT +
        LIGHT_LEVEL * LIGHT_DISTANCE_Y + USAGE_TEXT_SCOPE_HEIGHT * 2);
        memoryUsageBk.createRect(rectBk, Color.BLACK, Color.TRANSPARENT,
        Paint.Style.FILL_AND_STROKE);
        rlv.addView(memoryUsageBk);

        memoryTitleText = new ICNTextView(this);
        memoryTitleText.setWindowRange(x, y, LIGHT_WIDTH + LIGHT_DISTANCE_X +
        USAGE_SCOPE_EDGE * 2, USAGE_TITLE_SCOPE_HEIGHT);
        memoryTitleText.setText("Memory Usage");
        memoryTitleText.setTextSize(screen_height * 26 / 480);
        memoryTitleText.setTextColor(Color.YELLOW);
        memoryTitleText.setBackgroundColor(Color.TRANSPARENT);
        rlv.addView(memoryTitleText);

        for (int i = 0; i < LIGHT_LEVEL; i++) {
            memoryUsageGUIList.add(new ICNRegionView(this));
            Rect rectLeft = new Rect(USAGE_SCOPE_EDGE + x,
            USAGE_TITLE_SCOPE_HEIGHT + y + i * LIGHT_DISTANCE_Y, USAGE_SCOPE_EDGE + x +
            LIGHT_WIDTH, USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + i *
            LIGHT_DISTANCE_Y);
            memoryUsageGUIList.get(2 * i).createRect(rectLeft,
            Color.GREEN, Color.TRANSPARENT, Paint.Style.FILL_AND_STROKE);
            rlv.addView(memoryUsageGUIList.get(2 * i));

            memoryUsageGUIList.add(new ICNRegionView(this));
            Rect rectRight = new Rect(USAGE_SCOPE_EDGE + x +
            LIGHT_DISTANCE_X, USAGE_TITLE_SCOPE_HEIGHT + y + i * LIGHT_DISTANCE_Y,
            USAGE_SCOPE_EDGE + x + LIGHT_WIDTH + LIGHT_DISTANCE_X,
            USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + i * LIGHT_DISTANCE_Y);
            memoryUsageGUIList.get(2 * i + 1).createRect(rectRight,
            Color.GREEN, Color.TRANSPARENT, Paint.Style.FILL_AND_STROKE);
            rlv.addView(memoryUsageGUIList.get(2 * i + 1));
        }

        memoryUsageText = new ICNTextView(this);
        memoryUsageText.setWindowRange(USAGE_SCOPE_EDGE + x,
        USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + LIGHT_LEVEL * LIGHT_DISTANCE_Y,
        USAGE_TEXT_SCOPE_WIDTH, USAGE_TEXT_SCOPE_HEIGHT);
        memoryUsageText.setTextSize(screen_height * 30 / 480);
        memoryUsageText.setTextColor(Color.GREEN);
        memoryUsageText.setBackgroundColor(Color.TRANSPARENT);
        rlv.addView(memoryUsageText);

        memorySizeText = new ICNTextView(this);
        memorySizeText.setWindowRange(USAGE_SCOPE_EDGE + x,
        USAGE_TITLE_SCOPE_HEIGHT + y + LIGHT_HEIGHT + LIGHT_LEVEL * LIGHT_DISTANCE_Y
        + USAGE_TEXT_SCOPE_HEIGHT, USAGE_TEXT_SCOPE_WIDTH, USAGE_TEXT_SCOPE_HEIGHT);
    }

```

```

memorySizeText.setTextSize(screen_height * 25 / 480);
memorySizeText.setTextColor(Color.GREEN);
memorySizeText.setBackgroundColor(Color.TRANSPARENT);
rlv.addView(memorySizeText);
}

public void updateCPUUsage(int usage) {
    if (usage < 0 || usage > 100)
        return;

    int idle = 100 - usage;
    for (int i = 0; i < LIGHT_LEVEL; i++) {
        if (i < idle / 5) {
            cpuUsageGUIList.get(2 * i).alphaRefresh(50);
            cpuUsageGUIList.get(2 * i + 1).alphaRefresh(50);
            //Log.v("updateCPUUsage", Integer.toString(i));
        }
        else
        {
            cpuUsageGUIList.get(2 * i).alphaRefresh(100);
            cpuUsageGUIList.get(2 * i + 1).alphaRefresh(100);
        }
    }
    cpuUsageText.setText(Integer.toString(usage) + " %");
}

public void updateMemoryUsage(int usage) {
    if (usage < 0 || usage > 100)
        return;

    int idle = 100 - usage;
    for (int i = 0; i < LIGHT_LEVEL; i++) {
        if (i < idle / 5) {
            memoryUsageGUIList.get(2 * i).alphaRefresh(50);
            memoryUsageGUIList.get(2 * i + 1).alphaRefresh(50);
            //Log.v("updateMemoryUsage", Integer.toString(i));
        }
        else
        {
            memoryUsageGUIList.get(2 * i).alphaRefresh(100);
            memoryUsageGUIList.get(2 * i + 1).alphaRefresh(100);
        }
    }
    memoryUsageText.setText(Integer.toString(usage) + " %");
}

public void updateUsedMemorySize(float memsize, ICN_MEM_UNIT mu) {
    memorySizeText.setText(Float.toString(memsize) + " " + mu.toString());
}

private float readCPUUsage() {
    try {
        RandomAccessFile reader = new RandomAccessFile("/proc/stat", "r");
        String load = reader.readLine();

        String[] toks = load.split("\\s+");

```

```

        long idle1 = Long.parseLong(toks[4]);
        long cpu1 = Long.parseLong(toks[1]) + Long.parseLong(toks[2]) +
Long.parseLong(toks[3])
            + Long.parseLong(toks[5]) + Long.parseLong(toks[6]) +
Long.parseLong(toks[7]);

        try {
            Thread.sleep(500);
        } catch (Exception e) {}

        reader.seek(0);
        load = reader.readLine();
        reader.close();

        toks = load.split("\\s+");

        long idle2 = Long.parseLong(toks[4]);
        long cpu2 = Long.parseLong(toks[1]) + Long.parseLong(toks[2]) +
Long.parseLong(toks[3])
            + Long.parseLong(toks[5]) + Long.parseLong(toks[6]) +
Long.parseLong(toks[7]);

        return (float)(cpu2 - cpu1) * 100 / ((cpu2 + idle2) - (cpu1 +
idle1));

    } catch (IOException ex) {
        ex.printStackTrace();
    }

    return 0;
}

private float readMEMUsage() {
    try {
        RandomAccessFile reader = new RandomAccessFile("/proc/meminfo",
"r");

        String load = reader.readLine();

        String[] toks = load.split("\\s+");

        long mem_total = Long.parseLong(toks[1]);

        load = reader.readLine();
        reader.close();

        toks = load.split("\\s+");

        long mem_free = Long.parseLong(toks[1]);

        Mem_Used = mem_total - mem_free;

        return (float)(Mem_Used) * 100 / mem_total;

    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

```

```

        return 0;
    }

    public void resourceusageExitCallback(){
        unbindService();
        super.onDestroy();
        finish();
    }
}

```

API:

1. **void ICNRegionView::createCircle(float x, float y, float r, int color, int bkcolor, Paint.Style style)**

To create a circle view.

1. *float x: the x-coordinate of the center of the circle to be drawn*
2. *float y: the y-coordinate of the center of the circle to be drawn*
3. *int r: the radius of the circle to be drawn*
4. *int color: the color of the circle*
5. *Paint.Style style: three styles you can set.*

◆ *FILL*

◆ *STROKE*

◆ *FILL_AND_STROKE*

Please refer to the android official website for more information:

<http://developer.android.com/reference/android/graphics/Paint.Style.html>

Appendix B: NSD API Categories

The Smart Display API is a set of interfaces that describes the programming instructions provided by Android for accessing software applications.

API categories are major functional categories defined by Java classes. They are as follows:

Elevator, Graphical Drawing, Media Player, Ticker, Clock, Drag And Drop Feature and Chart etc.

Android apk and jar files

Android application package file (APK) is the [file format](#) used to distribute and install [application software](#) and [middleware](#) onto the Google's [Android](#) operating system. We provide an apk file called ICNMain.apk to hold of all the sample codes written in Java programming languages you need to run your Android device.

JAR (for Java Archive) is an archive file format typically used to aggregate many Java class files into one file to distribute libraries. Our JAR include many libraries such as ICNService, ICNImageButton, ICNDraw, ICNChart, ICNClock, ICNTicker, ICNMediaPlayer and ICNDragAndDrop. Every library here is an android library project. You can manage such projects from Eclipse with ADT.

Please refer to android official link.

<http://developer.android.com/guide/developing/projects/projects-eclipse.html>

Events

An event represents all activity that goes on between the user and the application. The java interfaces we provide to handle events for you are as follows:

- icnexus.smartdisplay.ICNUserData(ICNService library project)
- icnexus.smartdisplay.ICNUserString(ICNService library project)

These two events we provide handle user data and user text. You can **send your data or text to your smart display** using our predefined commands. Once the commands are received by your smart display service, it will notify the relative application of the callback event you have implemented before.

- icnexus.smartdisplay.ICNImageButtonCB(ICNImageButtonMgr library project)
- When you want to handle the button-down or button-up event, you can extend ICNImageButton and then implement your ICNImageButtonCB

- icnexus.smartdisplay.ICNDropZoneEventsListener(ICNDragAndDrop library project)
If you want to send user data after dropping your object, you can call the OnDropped() function of the SampleUserDragAndDrop.java file in the ICNMain project. You can call the ***sendUserData()*** fuction ***to send user data to your smart display.***
- icnexus.smartdisplay.ICNVideoPlayerEvent(ICNService library project)

The events for the video player on your interface can be handled using the callback functions as shown below:

1. playCallback() :
When you send the “play video” command to your smart display, this function will be called to play the video.
2. pauseCallback() :
When you send the “pause video” command to your smart display, this function will be called to pause the video,
3. stopCallback() :
When you send the “stop video” command to smart display, this function will be called to stop the video.

Services

- *We provide some services to receive or send user commands and they are as follows:*
 1. ICNCOMPortService: For RS232/RS485
 2. ICNNetworkService: For Network
- *Services must be started before you use your COM port or network.:*
- *How to start the service:*
 - i. Start COM Port Service by GUI:
Click the “COM Port Service Controller” icon



- ii. Start COM Port Service by the Console Command:

```
am startservice -n icnexus.smartdisplay/.service.ICNCOMPortService
```

- iii. *Start Network Service by GUI:*
Click the “Network Service Controller” icon



- iv. *Start Network Service by the Console Command:*
`am startservice -n icnexus.smartdisplay/.service.ICNNetworkService`

Sending and Receiving Commands via the Network

- *Open your Android Virtual Device Manager and then launch ICNCmdEmulator:*
- *Set the network IP of your smart display device and App ID. Then, send your commands*



NSD Command Definition

1. UART default Parameters

The default parameters of UART are:

Baudrate: 115200 bps

Band Rate Stop Bits: 1

Parity: Off

Handshake: None

2. Command Bytes:

Start byte: 1 byte

Command byte: 3byte

First byte is for special function. (SP)

Second byte is for command category. (CA)

Third byte is for main command. (MA)

Data byte: 1byte

End Bytes: 3byte

3. Command List:

User APP Command	Start Byte	Command Bytes	Data Bytes	End Bytes
User Elevator System 1. send command to smart display to launch application	0xFA	0x00 0x00 0x01	0x00	0xDD 0x89 0xBB
2. receive data from smart display (e.g. when user pushes button down # 6)	0xFB	0x00 0x00 0x01	0x06	0xDD 0x89 0xBB
3. send button-up command for specified button (after user arrives sixth floor)	0xFA	0x00 0x00 0x02	0x06	0xDD 0x89 0xBB

User Graphical Drawing UI	0xFA	0x00 0x01 0x01	0x00	0xDD 0x89 0xBB
User Digital Signage	0xFA	0x00 0x02 0x01	0x00	0xDD 0x89 0xBB

Send User Data Bytes To Smart Display Data Stream (16 bytes)	0xFA	0x01 0xFF 0x00	0x10 [16 bytes]	0xDD 0x89 0xBB
Data Stream (32 bytes)	0xFA	0x01 0xFF 0x00	0x20 [32 bytes]	0xDD 0x89 0xBB
Data Stream (256 bytes)	0xFA	0x01 0xFF 0x01	0x00 [256 bytes]	0xDD 0x89 0xBB
Send User Data Bytes Out Smart Display (16 bytes)	0xFB	0x01 0xFF 0x00	0x10 [16 bytes]	0xDD 0x89 0xBB

ICN Demo Command	Start Byte	Command Bytes	Data Byte	End Bytes
Elevator System 1. send command to smart display to launch application (e.g. Sansus font)	0xFA	0xD0 0x00 0x01	0x00	0xDD 0x89 0xBB
2. receive data from smart display (e.g. when user pushes button down #6)	0xFB	0xD0 0x00 0x01	0x06	0xDD 0x89 0xBB
3. send button-up command for specified button (e.g. after user arrives 6F)	0xFA	0xD0 0x00 0x02	0x06	0xDD 0x89 0xBB
Elevator System 1. send command to smart display to launch application (e.g. Acens font)	0xFA	0xD0 0x00 0x01	0x01	0xDD 0x89 0xBB

Graphical Drawing UI	0xFA	0xD0 0x01 0x01	0x00	0xDD 0x89 0xBB
----------------------	------	----------------	------	----------------

Application with Drag and Drop UI	0xFA	0xD0 0x02 0x01	0x00	0xDD 0x89 0xBB
-----------------------------------	------	----------------	------	----------------

Digital Signage	0xFA	0xD0 0x03 0x01	0x00	0xDD 0x89 0xBB
1. Launch Digital Signage				
2. Play/Resume Video	0xFA	0xD0 0x03 0x02	0x00	0xDD 0x89 0xBB
3. Pause Video	0xFA	0xD0 0x03 0x03	0x00	0xDD 0x89 0xBB
4. Stop Video	0xFA	0xD0 0x03 0x04	0x00	0xDD 0x89 0xBB
5. Video Forward	0xFA	0xD0 0x05 0x05	0x00	0xDD 0x89 0xBB
6. Video Backward	0xFA	0xD0 0x05 0x06	0x00	0xDD 0x89 0xBB
7. Next Video	0xFA	0xD0 0x05 0x07	0x00	0xDD 0x89 0xBB
8. Prev Video	0xFA	0xD0 0x05 0x08	0x00	0xDD 0x89 0xBB

Chart	0xFA	0xD0 0x04 0x01	0x00	0xDD 0x89 0xBB
-------	------	----------------	------	----------------

Media Player	0xFA	0xD0 0x05 0x01	0x00	0xDD 0x89 0xBB
1. Launch Video				
2. Exit Video	0xFA	0xD0 0x05 0x04	0x00	0xDD 0x89 0xBB
3. Play/Resume Video (after Pause)	0xFA	0xD0 0x05 0x02	0x00	0xDD 0x89 0xBB
4. Pause Video (after Play)	0xFA	0xD0 0x05 0x03	0x00	0xDD 0x89 0xBB
5. Video Forward	0xFA	0xD0 0x05 0x05	0x00	0xDD 0x89 0xBB
6. Video Backward	0xFA	0xD0 0x05 0x06	0x00	0xDD 0x89 0xBB
7. Next Video	0xFA	0xD0 0x05 0x07	0x00	0xDD 0x89 0xBB
8. Prev Video	0xFA	0xD0 0x05 0x08	0x00	0xDD 0x89 0xBB
9. Add New File Path From SD/DCIM	0xFA	0x02 0x05 0x00	0x?? (string len)	0x41 0x46 0x4B

File Browser 1. Launch FB 2. Exit File Browser 3. Update Playlist From: a. SD Card b. OHCI USB Flash 4. Update Video Playlist Update Audio Playlist Update Image Playlist	0xFA	0xD0 0x06 0x01	0x00	0xDD 0x89 0xBB
	0xFA	0xD0 0x06 0x02	0x00	0xDD 0x89 0xBB
	0xFA	0x02 0x06 0x00	0x02	0x53 0x44
	0xFA	0x02 0x06 0x00	0x04	0x4F 0x48 0x43 0x49
	0xFA	0xD0 0x06 0x03	0x00	0xDD 0x89 0xBB
	0xFA	0xD0 0x06 0x04	0x00	0xDD 0x89 0xBB
	0xFA	0xD0 0x06 0x05	0x00	0xDD 0x89 0xBB

Resource Usage GUI 1. Launch Resource Usage 2. Exit Resource Usage	0xFA	0xD0 0x07 0x01	0x00	0xDD 0x89 0xBB
	0xFA	0xD0 0x07 0x02	0x00	0xDD 0x89 0xBB

Device ID 1. Get ID - get 2 bytes ID 2. Set ID - give 2 bytes value	0xFA	0xD0 0x7F 0x01	0x00	0xDD 0x89 0xBB
	0xFA	0x01 0x7F 0x00	0x02	0x00 0x00

i. User Application

Button Control

1. send command to smart display to launch application

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0x00 0x00 0x01	0x00	0xDD 0x89 0xBB

2. receive data from smart display

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFB	0x00 0x00 0x01	0x06	0xDD 0x89 0xBB

Data bytes : button id clicked by user

(Its value is between 1 to 255)

3. send button-up command for specified button id to smart display

when button's status needs to be changed. (in this case, the button id is 6)

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0x00 0x00 0x02	0x06	0xDD 0x89 0xBB

User Graphical Drawing UI

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0x00 0x01 0x01	0x00	0xDD 0x89 0xBB

User Digital Signage

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0x01 0x02 0x01	0x00	0xDD 0x89 0xBB

ii. For ICN Demo

1. ICN Elevator System Demo

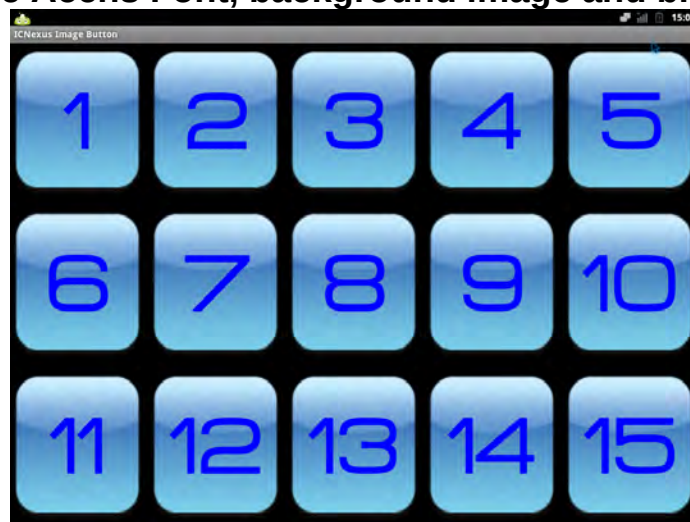
- Just set SP byte to 0x0D

Start Byte	Command Byte	Data Bytes	End Bytes
0xFA	0xD0 0x00 0x01	0x00	0xDD 0x89 0xBB

◆ Use Sansus Font, blue background image and blue text:



- ◆ Use Acens Font, background image and blue text:



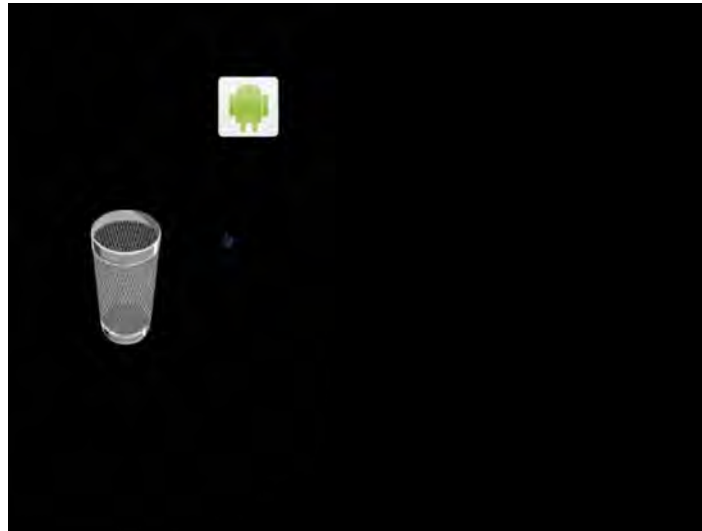
2. ICN Graphical Drawing UI Demo

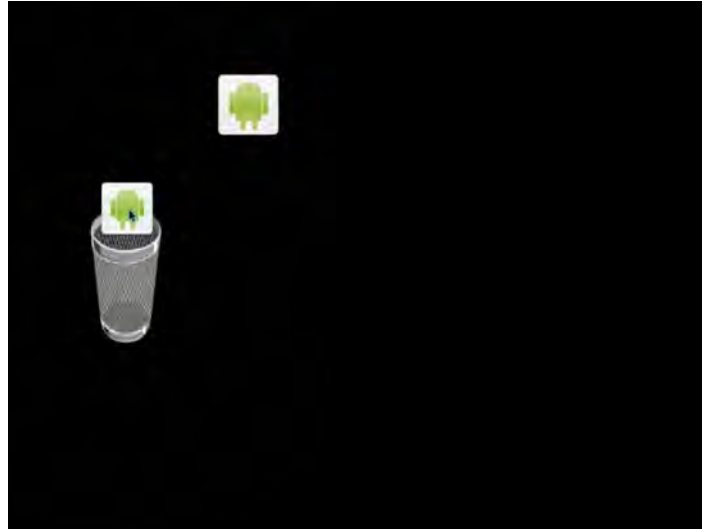
Start Byte	Command Byte	Data Bytes	End Bytes
0xFA	0xD0 0x01 0x01	0x00	0xDD 0x89 0xBB



3. Drag and Drop UI Demo

Start Byte	Command Byte	Data Bytes	End Bytes
0xFA	0xD0 0x02 0x01	0x00	0xDD 0x89 0xBB



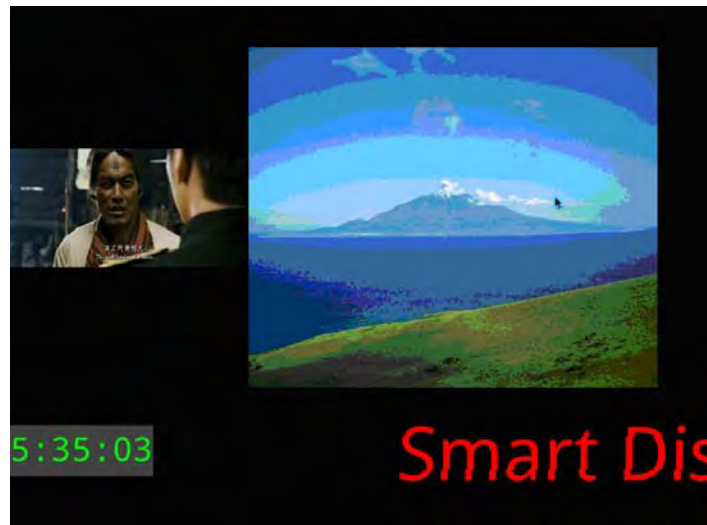


4. ICN Digital Signage Demo

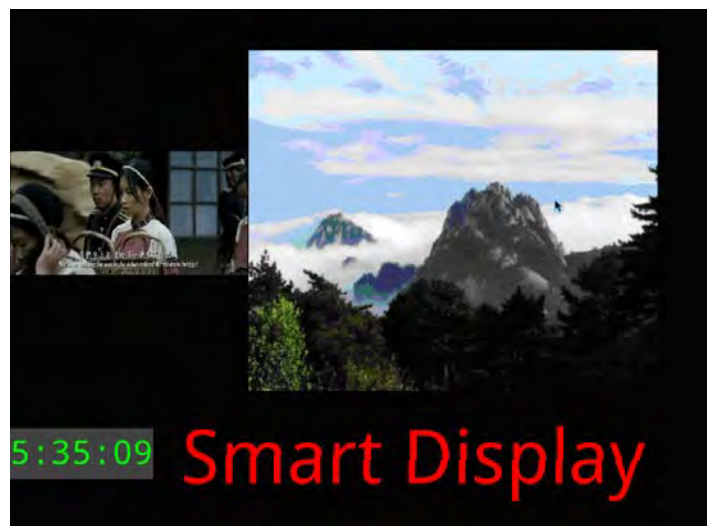
Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0xD0 0x03 0x01	0x00	0xDD 0x89 0xBB

A Digital Signage include clock, ticker, Image Player ◦

Time 1:



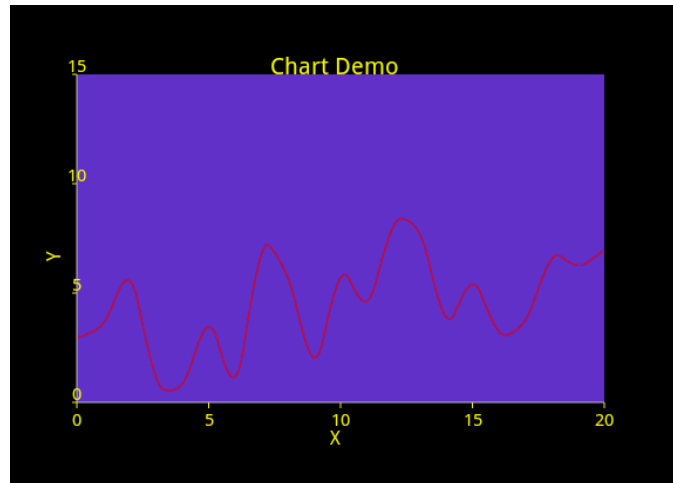
Time 2:



5. ICN Chart Demo

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0xD0 0x04 0x01	0x00	0xDD 0x89 0xBB

Draw a Chart diagram on screen.



6. ICN Media Player Demo

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0xD0 0x05 0x01	0x00	0xDD 0x89 0xBB

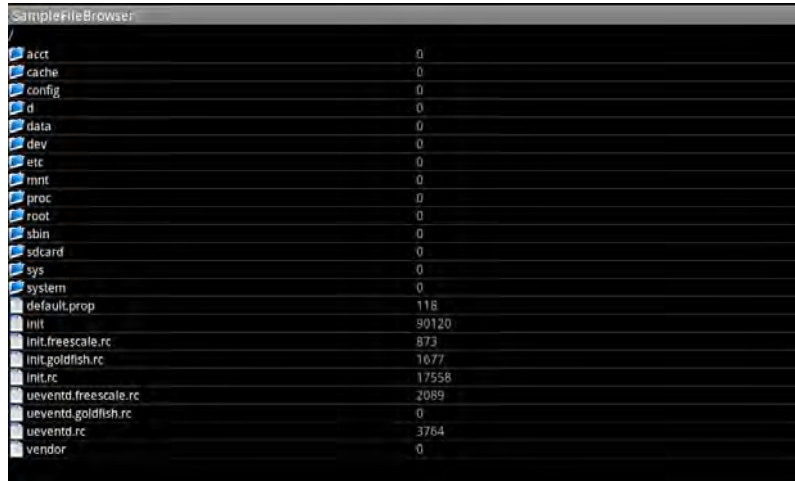
A Media Player screenshot:



7. ICN File Browser Demo

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0xD0 0x06 0x01	0x00	0xDD 0x89 0xBB

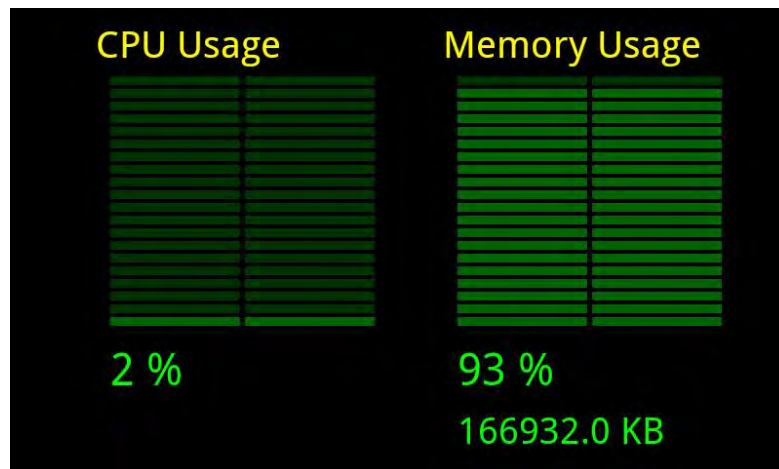
A file browser UI screenshot:



8. ICN Resource Usage GUI

Start Byte	Command Bytes	Data Bytes	End Bytes
0xFA	0xD0 0x07 0x01	0x00	0xDD 0x89 0xBB

A file browser UI screenshot:



- For Network Command Emulator

ICNCmdEmulator

Server IP : 192.168.11.2

Port : 20000

Send Command FAD0000100DD89 BB

App ID: 00

☐ Demo App

Send User Data (Hex) 0102030405060708 910111213141516

Send User Text (UTF-8) NSD Demo.....

Message from Smart Display